Beamlib

3.0.1

Generated by Doxygen 1.9.1

Chapter 1

Main Page

Author

Dr Terry Barnaby

Version

3.0.0

Date

2022-11-22

1.1 Introduction

The Beamlib C++ class library provides a system portable base library for developing real-time and other applications with multi-processor and multi-host support. It was initially started in the late 1980?s to add Smalltalk like constructions/components to the emerging C++ language for use within Beam for real-time data processing and embedded system uses. Over the years it has been extended as needed to support wildly differing projects.

The Beamlib system has the following features:

- Simple Object Orientated development.
- Simple class library for Strings, Lists, Arrays, Dictionaries, Network access etc.
- Support for multi-threaded applications with Mutex Objects etc.
- Usable from C++ and Python with wrapper.
- IDL based object creation tool allows easy creation of C++ and Python objects from IDL language with RPC access across networks.
- IDL provides the ability to create SQL database schema automatically.
- Database access that allows Objects to be stored.
- BOAP (Beam Object Access Protocol) provides a simple, low overhead protocol, that allows access to remote objects using an RPC mechanism.

- Database access via a layer that allows simultaineous access to different database systems including MYSQL and BEAM BDEV native object database.
- Simple HTTP/HTML WEB service support for application embedding.

The Beamlib class library can be installed from the following RPM packages:

- beamlib-lib: Runtime shared libraries.
- beamlib-utils: Runtime tools.
- beamlib-devel: Development include files and shared and static libraries.
- beamlib-doc: Documentation.

1.2 Components

The Beamlib system is split into the following libraries:

- Base: This is the base class library containing the base 'C++' classes.
- Http: This provides HTTP/HTML classes
- Widgets: This provides Qt widgets to be used on top of the standard Qt widgets.
- Gui1: This provides a simple GUI library for embedded systems.

The Beamlib system uses a few utilitry programs:

- bidl: This provides an IDL description file to C++ class generator.HTTP/HTML classes
- boapns: This provides a BOAP name server providing a name top object mapping system
- boapnsc: This provides a BOAP name server client for testing

For an overview and usage details please see the BeamlibApiManual.

1.3 and License

Beam Ltd holds the copyright of the Beamlib library. We provide it under the GNU GPLv3 General Public License version 3.0 open source licence for use by others, see LICENSE_GPLv3.txt. For projects Beam Ltd is involved in with our clients and for commercial use the software is available under other licenses including the LGPL license. Contact Beam Ltd for details.

1.4 API Examples

1.5 Examples

Some simple client examples are listed below:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Boapns Namespace Reference

Classes

- class Boapns
- class BoapEntry

Variables

- const BUInt32 apiVersion = 0

6.1.1 Variable Documentation

6.1.1.1 apiVersion

```
const BUInt32 Boapns::apiVersion = 0
```

Chapter 7

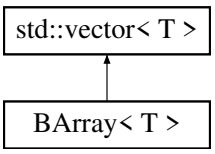
Class Documentation

7.1 BArray< T > Class Template Reference

Template based Array class.

`#include <BArray.h>`

Inheritance diagram for BArray< T >:



Public Types

- typedef int(* SortFunc) (T &a, T &b)
  Prototype for sorting function.

Public Member Functions

- BArray ()
- BArray (BSize size, T value=T())
- BArray (const BArray &array)
- BUInt number () const
- void append (const T &value)
- void append (const BArray< T > &array)
- void insert (BUInt pos, const T &value)
- void del (BUInt pos, BUInt num=1)
- T & rear ()
- void sort ()

7.1.1 Detailed Description

template<class T>
class BArray< T >

Template based Array class.

The BArray class is a simple contiguous in memory list of objects. It is used to store an ordered list of any type/class of objects. It is based on the Standard C++ library vector class and has all of the functionality of that class.

7.1.2 Member Typedef Documentation

7.1.2.1 SortFunc

```
template<class T >
typedef int(* BArray< T >::SortFunc) (T &a, T &b)
```

Prototype for sorting function.

7.1.3 Constructor & Destructor Documentation

7.1.3.1 BArray() [1/3]

```
template<class T >
BArray< T >::BArray ( )  [inline]
```

7.1.3.2 BArray() [2/3]

```
template<class T >
BArray< T >::BArray (
            BSize size,
            T value = T() )  [inline]
```

7.1.3.3 BArray() [3/3]

```
template<class T >
BArray< T >::BArray (
            const BArray< T > & array )  [inline]
```

7.1.4 Member Function Documentation

7.1.4.1 number()

```
template<class T >
BUInt BArray< T >::number ( ) const  [inline]
```

7.1.4.2 append() [1/2]

```
template<class T >
void BArray< T >::append (
            const T & value )  [inline]
```

7.1.4.3 append() [2/2]

```
template<class T >
void BArray< T >::append (
            const BArray< T > & array )
```

7.1.4.4 insert()

```
template<class T >
void BArray< T >::insert (
            BUInt pos,
            const T & value )  [inline]
```

7.1.4.5 del()

```
template<class T >
void BArray< T >::del (
            BUInt pos,
            BUInt num = 1 )  [inline]
```

7.1.4.6 rear()

```
template<class T >
T& BArray< T >::rear ( )  [inline]
```

7.1.4.7 sort()

```
template<class T >
void BArray< T >::sort ( )  [inline]
```

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

- BArray.h

7.2 BAtomic< Type > Class Template Reference

BAtomic class increments/decrements different integer types.

```
#include <BAtomic.h>
```

Public Member Functions

- BAtomic (Type value=0)
- Type getValue () const
- Type add (long value)
- Type operator++ (int)
- Type operator++ ()
- Type operator-- (int)
- Type operator-- ()
- operator Type () const

7.2.1 Detailed Description

template<class Type>
class BAtomic< Type >

BAtomic class increments/decrements different integer types.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BAtomic()

```
template<class Type >
BAtomic< Type >::BAtomic (
            Type value = 0 )  [inline]
```

7.2.3 Member Function Documentation

7.2.3.1 getValue()

```
template<class Type >
Type BAtomic< Type >::getValue ( ) const  [inline]
```

7.2.3.2 add()

```
template<class Type >
Type BAtomic< Type >::add (
            long value )  [inline]
```

7.2.3.3 operator++() [1/2]

```
template<class Type >
Type BAtomic< Type >::operator++ (
            int  )  [inline]
```

7.2.3.4 operator++() [2/2]

```
template<class Type >
Type BAtomic< Type >::operator++ ( )  [inline]
```

7.2.3.5 operator--() [1/2]

```
template<class Type >
Type BAtomic< Type >::operator-- (
            int  )  [inline]
```

7.2.3.6 operator--() [2/2]

```
template<class Type >
Type BAtomic< Type >::operator-- ( )  [inline]
```

7.2.3.7 operator Type()

```
template<class Type >
BAtomic< Type >::operator Type ( ) const  [inline]
```

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

- BAtomic.h

7.3 BAtomicCount Class Reference

BAtomicCount class.

```
#include <BAtomicCount.h>
```

Public Member Functions

- BAtomicCount (long value=0)
- long getValue () const
- long add (long value)
- long operator++ (int)
- long operator++ ()
- long operator-- (int)
- long operator-- ()
- operator long () const

7.3.1 Detailed Description

BAtomicCount class.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BAtomicCount()

```
BAtomicCount::BAtomicCount (
            long value = 0 )  [inline]
```

7.3.3 Member Function Documentation

7.3.3.1 getValue()

```
long BAtomicCount::getValue ( ) const  [inline]
```

7.3.3.2 add()

```
long BAtomicCount::add (
            long value )  [inline]
```

7.3.3.3 operator++() [1/2]

```
long BAtomicCount::operator++ (
            int  )  [inline]
```

7.3.3.4 operator++() [2/2]

```
long BAtomicCount::operator++ ( )  [inline]
```

7.3.3.5 operator--() [1/2]

```
long BAtomicCount::operator-- (
            int  )  [inline]
```

7.3.3.6 operator--() [2/2]

```
long BAtomicCount::operator-- ( )  [inline]
```

7.3.3.7 operator long()

```
BAtomicCount::operator long ( ) const  [inline]
```

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

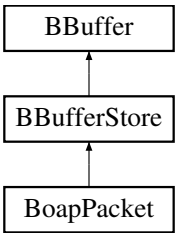
- BAtomicCount.h

7.4 BBuffer Class Reference

Create and manipulate a variable sized byte data buffer.

```
#include <BBuffer.h>
```

Inheritance diagram for BBuffer:



Public Member Functions

- BBuffer (BUInt size=0)
- ~BBuffer ()
- int setSize (BUInt32 size)

  Sets the bufer size.
- int setData (const void *data, BUInt32 size)

  Sets buffer data resized to contain the data.
- int writeData (BUInt32 pos, const void *data, BUInt32 size)

  Writes data into buffer from offset pos.
- char * data ()

  The data.
- BUInt32 size ()

  Size of the buffer in bytes.
- int resize (BUInt32 size)

  Alternative to setSize()

Protected Attributes

- BUInt32 odataSize
- char * odata
- BUInt32 osize

7.4.1 Detailed Description

Create and manipulate a variable sized byte data buffer.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 BBuffer()

```
BBuffer::BBuffer (
            BUInt size = 0 )
```

7.4.2.2 ~BBuffer()

```
BBuffer::~BBuffer ( )
```

7.4.3 Member Function Documentation

7.4.3.1 setSize()

```
int BBuffer::setSize (
            BUInt32 size )
```

Sets the bufer size.

7.4.3.2 setData()

```
int BBuffer::setData (
            const void * data,
            BUInt32 size )
```

Sets buffer data resized to contain the data.

7.4.3.3 writeData()

```
int BBuffer::writeData (
            BUInt32 pos,
            const void * data,
            BUInt32 size )
```

Writes data into buffer from offset pos.

7.4.3.4 data()

```
char * BBuffer::data ( )
```

The data.

7.4.3.5 size()

```
BUInt32 BBuffer::size ( )
```

Size of the buffer in bytes.

7.4.3.6 resize()

```
int BBuffer::resize (
            BUInt32 size )  [inline]
```

Alternative to setSize()

7.4.4 Member Data Documentation

7.4.4.1 odataSize

```
BUInt32 BBuffer::odataSize  [protected]
```

7.4.4.2 odata

```
char* BBuffer::odata  [protected]
```

7.4.4.3 osize

```
BUInt32 BBuffer::osize  [protected]
```

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

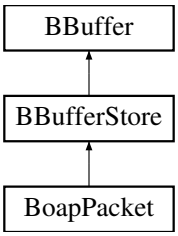
- BBuffer.h
- BBuffer.cpp

7.5 BBufferStore Class Reference

Create and manipulate a variable sized byte data buffer. Has functions to store and retrieve basic and extended types/classes in the binary buffer.

```
#include <BBuffer.h>
```

Inheritance diagram for BBufferStore:



Public Member Functions

- BBufferStore (BUInt size=0, int swapBytes=BBigEndian)
- ~BBufferStore ()
- BUInt32 getPos ()
- void setPos (BUInt32 pos)
- BString getHexString ()
- void setHexString (BString s)
- int push (BInt8 v)
- int push (BUInt8 v)
- int push (BInt16 v)
- int push (BUInt16 v)
- int push (BInt32 v)
- int push (BUInt32 v)
- int push (BInt64 v)
- int push (BUInt64 v)
- int push (BFloat32 v)
- int push (BFloat64 v)
- int push (const BString &v)
- int push (const BError &v)
- int push (const BTimeStamp &v)
- int push (const BComplex &v)
- int push (BUInt32 nBytes, const void *data, const char *swapType="1")
- int pop (BInt8 &v)
- int pop (BUInt8 &v)

- int pop (BInt16 &v)
- int pop (BUInt16 &v)
- int pop (BInt32 &v)
- int pop (BUInt32 &v)
- int pop (BInt64 &v)
- int pop (BUInt64 &v)
- int pop (BFloat32 &v)
- int pop (BFloat64 &v)
- int pop (BString &v)
- int pop (BError &v)
- int pop (BTimeStamp &v)
- int pop (BComplex &v)
- int pop (BUInt32 nBytes, void *data, const char *swapType="1")

Protected Attributes

- BUInt32 opos
- int oswapBytes

7.5.1 Detailed Description

Create and manipulate a variable sized byte data buffer. Has functions to store and retrieve basic and extended types/classes in the binary buffer.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 BBufferStore()

```
BBufferStore::BBufferStore (
            BUInt size = 0,
            int swapBytes = BBigEndian )
```

7.5.2.2 ~BBufferStore()

```
BBufferStore::~BBufferStore ( )
```

7.5.3 Member Function Documentation

7.5.3.1 getPos()

```
BUInt32 BBufferStore::getPos ( )
```

7.5.3.2 setPos()

```
void BBufferStore::setPos (
            BUInt32 pos )
```

7.5.3.3 getHexString()

```
BString BBufferStore::getHexString ( )
```

7.5.3.4 setHexString()

```
void BBufferStore::setHexString (
            BString s )
```

7.5.3.5 push() [1/15]

```
int BBufferStore::push (
            BInt8 v )
```

7.5.3.6 push() [2/15]

```
int BBufferStore::push (
            BUInt8 v )
```

7.5.3.7 push() [3/15]

```
int BBufferStore::push (
            BInt16 v )
```

7.5.3.8 push() [4/15]

```
int BBufferStore::push (
            BUInt16 v )
```

7.5.3.9 push() [5/15]

```
int BBufferStore::push (
            BInt32 v )
```

7.5.3.10 push() [6/15]

```
int BBufferStore::push (
            BUInt32 v )
```

7.5.3.11 push() [7/15]

```
int BBufferStore::push (
            BInt64 v )
```

7.5.3.12 push() [8/15]

```
int BBufferStore::push (
            BUInt64 v )
```

7.5.3.13 push() [9/15]

```
int BBufferStore::push (
            BFloat32 v )
```

7.5.3.14 push() [10/15]

```
int BBufferStore::push (
            BFloat64 v )
```

7.5.3.15 push() [11/15]

```
int BBufferStore::push (
            const BString & v )
```

7.5.3.16 push() [12/15]

```
int BBufferStore::push (
            const BError & v )
```

7.5.3.17 push() [13/15]

```
int BBufferStore::push (
            const BTimeStamp & v )
```

7.5.3.18 push() [14/15]

```
int BBufferStore::push (
            const BComplex & v )
```

7.5.3.19 push() [15/15]

```
int BBufferStore::push (
            BUInt32 nBytes,
            const void * data,
            const char * swapType = "1" )
```

7.5.3.20 pop() [1/15]

```
int BBufferStore::pop (
            BInt8 & v )
```

7.5.3.21 pop() [2/15]

```
int BBufferStore::pop (
            BUInt8 & v )
```

7.5.3.22 pop() [3/15]

```
int BBufferStore::pop (
            BInt16 & v )
```

7.5.3.23 pop() [4/15]

```
int BBufferStore::pop (
            BUInt16 & v )
```

7.5.3.24 pop() [5/15]

```
int BBufferStore::pop (
            BInt32 & v )
```

7.5.3.25 pop() [6/15]

```
int BBufferStore::pop (
            BUInt32 & v )
```

7.5.3.26 pop() [7/15]

```
int BBufferStore::pop (
            BInt64 & v )
```

7.5.3.27 pop() [8/15]

```
int BBufferStore::pop (
            BUInt64 & v )
```

7.5.3.28 pop() [9/15]

```
int BBufferStore::pop (
            BFloat32 & v )
```

7.5.3.29 pop() [10/15]

```
int BBufferStore::pop (
            BFloat64 & v )
```

7.5.3.30 pop() [11/15]

```
int BBufferStore::pop (
            BString & v )
```

7.5.3.31 pop() [12/15]

```
int BBufferStore::pop (
            BError & v )
```

7.5.3.32 pop() [13/15]

```
int BBufferStore::pop (
            BTimeStamp & v )
```

7.5.3.33 pop() [14/15]

```
int BBufferStore::pop (
            BComplex & v )
```

7.5.3.34 pop() [15/15]

```
int BBufferStore::pop (
            BUInt32 nBytes,
            void * data,
            const char * swapType = "1" )
```

7.5.4 Member Data Documentation

7.5.4.1 opos

```
BUInt32 BBufferStore::opos  [protected]
```

7.5.4.2 oswapBytes

```
int BBufferStore::oswapBytes  [protected]
```

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

- BBuffer.h
- BBuffer.cpp

7.6 BComms Class Reference

A base class for communications classes having a generic API.

`#include <BComms.h>`

Public Types

- enum Flush { FlushRead , FlushWrite , FlushReadWrite }

Public Member Functions

- BComms ()
- virtual ~BComms ()
- virtual BError init ()
- virtual void close ()
- virtual const char * name ()

  The name of this interface.
- virtual BUInt32 byteRate ()

  The byte rate of this interface.
- virtual BError setPacketMode (Bool packetMode)

  Set packet mode.
- virtual Bool packetMode ()

  Device is in packet mode.
- virtual BError setTimeout (BTimeout timeoutUs)

  Set communication timeout.
- virtual BError connect (const char *resource)

  Create a connection.
- virtual Bool isConnected ()
- virtual BError disconnect ()

  Disconnect.
- virtual void flush (Flush flush)
- virtual BUInt writeAvailable ()
- virtual BError write (const void *data, BUInt32 nBytes, BUInt32 &nTrans)=0
- virtual BError writeChunks (const BDataChunk *chunks, BUInt nChunks, BUInt32 &nTrans)
- virtual BUInt readAvailable ()
- virtual BError read (void *data, BUInt32 num, BUInt32 &nTrans)=0
- virtual BError wait (BUInt32 eventSet, BTimeout timeoutUs=BTimeoutForever, BUInt32 num=1)
- virtual void eventQueue (BEventQueue *eventQueue, BUInt32 event, BUInt32 eventSet, BUInt num=1)
- virtual void eventEnable (Bool on)

  Enable events to be sent.

Protected Attributes

- Bool oconnected
- Bool opacketMode
- BTimeout otimeout
- BEventQueue * oeventQueue
- Bool oeventEnabled
- BUInt32 oevent
- BUInt32 oeventSet
- BUInt oeventNum

7.6.1 Detailed Description

A base class for communications classes having a generic API.

7.6.2 Member Enumeration Documentation

7.6.2.1 Flush

```
enum BComms::Flush
```

Enumerator

FlushRead	
FlushWrite	
FlushReadWrite	

7.6.3 Constructor & Destructor Documentation

7.6.3.1 BComms()

```
BComms::BComms ( )
```

7.6.3.2 ~BComms()

```
BComms::~BComms ( )  [virtual]
```

7.6.4 Member Function Documentation

7.6.4.1 init()

```
BError BComms::init ( )  [virtual]
```

7.6.4.2 close()

```
void BComms::close ( )  [virtual]
```

7.6.4.3 name()

```
const char * BComms::name ( )  [virtual]
```

The name of this interface.

7.6.4.4 byteRate()

```
BUInt32 BComms::byteRate ( )  [virtual]
```

The byte rate of this interface.

7.6.4.5 setPacketMode()

```
BError BComms::setPacketMode (
            Bool packetMode )  [virtual]
```

Set packet mode.

7.6.4.6 packetMode()

```
Bool BComms::packetMode ( )  [virtual]
```

Device is in packet mode.

7.6.4.7 setTimeout()

```
BError BComms::setTimeout (
            BTimeout timeoutUs )  [virtual]
```

Set communication timeout.

7.6.4.8 connect()

```
BError BComms::connect (
            const char * resource )  [virtual]
```

Create a connection.

7.6.4.9 isConnected()

```
Bool BComms::isConnected ( )  [virtual]
```

7.6.4.10 disconnect()

```
BError BComms::disconnect ( )  [virtual]
```

Disconnect.

7.6.4.11 flush()

```
void BComms::flush (
            Flush flush )  [virtual]
```

7.6.4.12 writeAvailable()

```
BUInt BComms::writeAvailable ( )  [virtual]
```

7.6.4.13 write()

```
virtual BError BComms::write (
            const void * data,
            BUInt32 nBytes,
            BUInt32 & nTrans )  [pure virtual]
```

7.6.4.14 writeChunks()

```
BError BComms::writeChunks (
            const BDataChunk * chunks,
            BUInt nChunks,
            BUInt32 & nTrans )  [virtual]
```

7.6.4.15 readAvailable()

```
BUInt BComms::readAvailable ( )  [virtual]
```

7.6.4.16 read()

```
virtual BError BComms::read (
            void * data,
            BUInt32 num,
            BUInt32 & nTrans )  [pure virtual]
```

7.6.4.17 wait()

```
BError BComms::wait (
            BUInt32 eventSet,
            BTimeout timeoutUs = BTimeoutForever,
            BUInt32 num = 1 )  [virtual]
```

7.6.4.18 eventQueue()

```
void BComms::eventQueue (
            BEventQueue * eventQueue,
            BUInt32 event,
            BUInt32 eventSet,
            BUInt num = 1 )  [virtual]
```

7.6.4.19 eventEnable()

```
void BComms::eventEnable (
            Bool on )  [virtual]
```

Enable events to be sent.

7.6.5 Member Data Documentation

7.6.5.1 oconnected

```
Bool BComms::oconnected  [protected]
```

7.6.5.2 opacketMode

```
Bool BComms::opacketMode  [protected]
```

7.6.5.3 otimeout

```
BTimeout BComms::otimeout  [protected]
```

7.6.5.4 oeventQueue

```
BEventQueue* BComms::oeventQueue  [protected]
```

7.6.5.5 oeventEnabled

```
Bool BComms::oeventEnabled  [protected]
```

7.6.5.6 oevent

```
BUInt32 BComms::oevent  [protected]
```

7.6.5.7 oeventSet

```
BUInt32 BComms::oeventSet  [protected]
```

7.6.5.8 oeventNum

```
BUInt BComms::oeventNum  [protected]
```

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

- BComms.h
- BComms.cpp

7.7 BCond Class Reference

Thread safe conditional variable.

```
#include <BCond.h>
```

Public Member Functions

- BCond ()
- ~BCond ()
- int signal ()
- int wait ()
- int timedWait (int timeOutUs)

7.7.1 Detailed Description

Thread safe conditional variable.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 BCond()

```
BCond::BCond ( )
```

7.7.2.2 ~BCond()

```
BCond::~BCond ( )
```

7.7.3 Member Function Documentation

7.7.3.1 signal()

```
int BCond::signal ( )
```

7.7.3.2 wait()

```
int BCond::wait ( )
```

7.7.3.3 timedWait()

```
int BCond::timedWait (
            int timeOutUs )
```

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

- BCond.h
- BCond.cpp

7.8 BCondBool Class Reference

Thread conditional boolean.

```
#include <BCondInt.h>
```

Public Member Functions

- BCondBool ()
- ~BCondBool ()
- int set ()
  Set value. Wakes waiting.
- int clear ()
  Clear Value.
- int value ()
  Current value.
- int wait ()
  Wait until value is true.
- int timedWait (int timeOutUs)
  Wait until set, with timeout.
- operator int ()

7.8.1 Detailed Description

Thread conditional boolean.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 BCondBool()

```
BCondBool::BCondBool ( )
```

7.8.2.2 ~BCondBool()

```
BCondBool::~BCondBool ( )
```

7.8.3 Member Function Documentation

7.8.3.1 set()

```
int BCondBool::set ( )
```

Set value. Wakes waiting.

7.8.3.2 clear()

```
int BCondBool::clear ( )
```

Clear Value.

7.8.3.3 value()

```
int BCondBool::value ( )
```

Current value.

7.8.3.4 wait()

```
int BCondBool::wait ( )
```

Wait until value is true.

7.8.3.5 timedWait()

```
int BCondBool::timedWait (
            int timeOutUs )
```

Wait until set, with timeout.

7.8.3.6 operator int()

```
BCondBool::operator int ( )  [inline]
```

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

- BCondInt.h
- BCondInt.cpp

7.9 BCondInt Class Reference

Thread conditional value.

`#include <BCondInt.h>`

Public Member Functions

- BCondInt ()
- ~BCondInt ()
- void setValue (BInt value)

  Set the value. Wakes waiting.
- BInt value () const

  Current value.
- BInt increment (BInt v=1)

  Increment. Wakes waiting.
- BInt decrement (BInt v=1)

  Decrement. Wakes waiting.
- Bool waitMoreThanOrEqual (BInt v, Bool decrement=0, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is at least the value given.
- Bool waitLessThanOrEqual (BInt v, Bool increment=0, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is equal to or below the value given.
- Bool waitLessThan (BInt v, BTimeout timeoutUs=BTimeoutForever)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

7.9.1 Detailed Description

Thread conditional value.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 BCondInt()

```
BCondInt::BCondInt ( )
```

7.9.2.2 ~BCondInt()

```
BCondInt::~BCondInt ( )
```

7.9.3 Member Function Documentation

7.9.3.1 setValue()

```
void BCondInt::setValue (
            BInt value )
```

Set the value. Wakes waiting.

7.9.3.2 value()

```
BInt BCondInt::value ( ) const
```

Current value.

7.9.3.3 increment()

```
BInt BCondInt::increment (
            BInt v = 1 )
```

Increment. Wakes waiting.

7.9.3.4 decrement()

```
BInt BCondInt::decrement (
            BInt v = 1 )
```

Decrement. Wakes waiting.

7.9.3.5 waitMoreThanOrEqual()

```
Bool BCondInt::waitMoreThanOrEqual (
            BInt v,
            Bool decrement = 0,
            BTimeout timeoutUs = BTimeoutForever )
```

Wait until value is at least the value given.

7.9.3.6 waitLessThanOrEqual()

```
Bool BCondInt::waitLessThanOrEqual (
            BInt v,
            Bool increment = 0,
            BTimeout timeoutUs = BTimeoutForever )
```

Wait until value is equal to or below the value given.

7.9.3.7 waitLessThan()

```
Bool BCondInt::waitLessThan (
            BInt v,
            BTimeout timeoutUs = BTimeoutForever )
```

Wait until value is equal to or below the value given.

7.9.3.8 operator+=()

```
void BCondInt::operator+= (
            int v )  [inline]
```

Add to value. Wakes waiting.

7.9.3.9 operator-=()

```
void BCondInt::operator-= (
            int v )  [inline]
```

Subtract from value. Wakes waiting.

7.9.3.10 operator++()

```
void BCondInt::operator++ (
            int  )  [inline]
```

Increment value. Wakes waiting.

7.9.3.11 operator--()

```
void BCondInt::operator-- (
            int  )  [inline]
```

Decrement value. Wakes waiting.

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

- BCondInt.h
- BCondInt.cpp

7.10 BCondResource Class Reference

Resource lock.

`#include <BCondInt.h>`

Public Member Functions

- BCondResource ()
- ~BCondResource ()
- int lock (uint32_t timeOutUs=0)

  Lock the resource, will wait for all usage to be 0.
- int unlock ()

  Unlock the resource.
- int start (uint32_t timeOutUs=0)

  Start using the resource.
- int end ()

  Finish using the resource.
- int locked ()
- int inUse ()

7.10.1 Detailed Description

Resource lock.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 BCondResource()

```
BCondResource::BCondResource ( )
```

7.10.2.2 ~BCondResource()

```
BCondResource::~BCondResource ( )
```

7.10.3 Member Function Documentation

7.10.3.1 lock()

```
int BCondResource::lock (
            uint32_t timeOutUs = 0 )
```

Lock the resource, will wait for all usage to be 0.

7.10.3.2 unlock()

```
int BCondResource::unlock ( )
```

Unlock the resource.

7.10.3.3 start()

```
int BCondResource::start (
            uint32_t timeOutUs = 0 )
```

Start using the resource.

7.10.3.4 end()

```
int BCondResource::end ( )
```

Finish using the resource.

7.10.3.5 locked()

```
int BCondResource::locked ( )
```

7.10.3.6 inUse()

```
int BCondResource::inUse ( )
```

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

- BCondInt.h
- BCondInt.cpp

7.11 BCondValue Class Reference

Thread conditional value.

```
#include <BCondInt.h>
```

Public Member Functions

- BCondValue ()
- ~BCondValue ()
- void setValue (int value)

  Set the value. Wakes waiting.
- int value ()

  Current value.
- int increment (int v=1)

  Increment. Wakes waiting.
- int decrement (int v=1)

  Decrement. Wakes waiting.
- int waitMoreThanOrEqual (int v, int decrement=0, int timeOutUs=0)

  Wait until value is at least the value given.
- int waitLessThanOrEqual (int v, int increment=0, int timeOutUs=0)

  Wait until value is equal to or below the value given.
- int waitLessThan (int v, int timeOutUs=0)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

7.11.1 Detailed Description

Thread conditional value.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 BCondValue()

```
BCondValue::BCondValue ( )
```

7.11.2.2 ~BCondValue()

```
BCondValue::~BCondValue ( )
```

7.11.3 Member Function Documentation

7.11.3.1 setValue()

```
void BCondValue::setValue (
            int value )
```

Set the value. Wakes waiting.

7.11.3.2 value()

```
int BCondValue::value ( )
```

Current value.

7.11.3.3 increment()

```
int BCondValue::increment (
            int v = 1 )
```

Increment. Wakes waiting.

7.11.3.4 decrement()

```
int BCondValue::decrement (
            int v = 1 )
```

Decrement. Wakes waiting.

7.11.3.5 waitMoreThanOrEqual()

```
int BCondValue::waitMoreThanOrEqual (
            int v,
            int decrement = 0,
            int timeOutUs = 0 )
```

Wait until value is at least the value given.

7.11.3.6 waitLessThanOrEqual()

```
int BCondValue::waitLessThanOrEqual (
            int v,
            int increment = 0,
            int timeOutUs = 0 )
```

Wait until value is equal to or below the value given.

7.11.3.7 waitLessThan()

```
int BCondValue::waitLessThan (
            int v,
            int timeOutUs = 0 )
```

Wait until value is equal to or below the value given.

7.11.3.8 operator+=()

```
void BCondValue::operator+= (
            int v )  [inline]
```

Add to value. Wakes waiting.

7.11.3.9 operator-=()

```
void BCondValue::operator-= (
            int v )  [inline]
```

Subtract from value. Wakes waiting.

7.11.3.10 operator++()

```
void BCondValue::operator++ (
            int  )  [inline]
```

Increment value. Wakes waiting.

7.11.3.11 operator--()

```
void BCondValue::operator-- (
            int  )  [inline]
```

Decrement value. Wakes waiting.

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

- BCondInt.h
- BCondInt.cpp

7.12 BCondWrap Class Reference

Thread conditional unsigned 32 bit integer value that can wrap around.

```
#include <BCondInt.h>
```

Public Member Functions

- BCondWrap ()
- ~BCondWrap ()
- void setValue (uint32_t value)

  Set the value. Wakes waiting.
- uint32_t value ()

  Current value.
- uint32_t increment (uint32_t v=1)

  Increment. Wakes waiting.
- uint32_t decrement (uint32_t v=1)

  Decrement. Wakes waiting.
- int waitMoreThanOrEqual (uint32_t v, uint32_t decrement=0, uint32_t timeOutUs=0)

  Wait until value is at least the value given.
- int waitLessThanOrEqual (uint32_t v, uint32_t increment=0, uint32_t timeOutUs=0)

  Wait until value is equal to or below the value given.
- int waitLessThan (uint32_t v, uint32_t timeOutUs=0)

  Wait until value is equal to or below the value given.
- void operator+= (int v)

  Add to value. Wakes waiting.
- void operator-= (int v)

  Subtract from value. Wakes waiting.
- void operator++ (int)

  Increment value. Wakes waiting.
- void operator-- (int)

  Decrement value. Wakes waiting.

7.12.1 Detailed Description

Thread conditional unsigned 32 bit integer value that can wrap around.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 BCondWrap()

```
BCondWrap::BCondWrap ( )
```

7.12.2.2 ~BCondWrap()

```
BCondWrap::~BCondWrap ( )
```

7.12.3 Member Function Documentation

7.12.3.1 setValue()

```
void BCondWrap::setValue (
            uint32_t value )
```

Set the value. Wakes waiting.

7.12.3.2 value()

```
uint32_t BCondWrap::value ( )
```

Current value.

7.12.3.3 increment()

```
uint32_t BCondWrap::increment (
            uint32_t v = 1 )
```

Increment. Wakes waiting.

7.12.3.4 decrement()

```
uint32_t BCondWrap::decrement (
            uint32_t v = 1 )
```

Decrement. Wakes waiting.

7.12.3.5 waitMoreThanOrEqual()

```
int BCondWrap::waitMoreThanOrEqual (
            uint32_t v,
            uint32_t decrement = 0,
            uint32_t timeOutUs = 0 )
```

Wait until value is at least the value given.

7.12.3.6 waitLessThanOrEqual()

```
int BCondWrap::waitLessThanOrEqual (
            uint32_t v,
            uint32_t increment = 0,
            uint32_t timeOutUs = 0 )
```

Wait until value is equal to or below the value given.

7.12.3.7 waitLessThan()

```
int BCondWrap::waitLessThan (
            uint32_t v,
            uint32_t timeOutUs = 0 )
```

Wait until value is equal to or below the value given.

7.12.3.8 operator+=()

```
void BCondWrap::operator+= (
            int v )  [inline]
```

Add to value. Wakes waiting.

7.12.3.9 operator-=()

```
void BCondWrap::operator-= (
            int v )  [inline]
```

Subtract from value. Wakes waiting.

7.12.3.10 operator++()

```
void BCondWrap::operator++ (
            int  )  [inline]
```

Increment value. Wakes waiting.

7.12.3.11 operator--()

```
void BCondWrap::operator-- (
            int  )  [inline]
```

Decrement value. Wakes waiting.

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

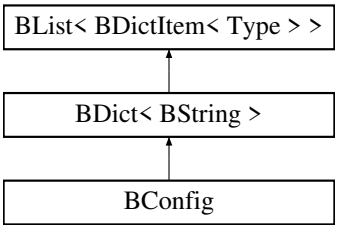
- BCondInt.h
- BCondInt.cpp

7.13 BConfig Class Reference

This class implements the configuration file access.

```
#include <BConfig.h>
```

Inheritance diagram for BConfig:



Public Member Functions

- BError open (BString fileName, BString mode="r")
- void close ()
- BError read ()
- BError write ()
- BString findValue (BString name)
- BString fileName ()

Additional Inherited Members

7.13.1 Detailed Description

This class implements the configuration file access.

7.13.2 Member Function Documentation

7.13.2.1 open()

```
BError BConfig::open (
            BString fileName,
            BString mode = "r" )
```

7.13.2.2 close()

```
void BConfig::close ( )
```

7.13.2.3 read()

```
BError BConfig::read ( )
```

7.13.2.4 write()

```
BError BConfig::write ( )
```

7.13.2.5 findValue()

```
BString BConfig::findValue (
            BString name )
```

7.13.2.6 fileName()

```
BString BConfig::fileName ( )
```

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

- BConfig.h
- BConfig.cpp

7.14 BDataChunk Class Reference

A chunk of data allowing writes of multiple chunks of segmented data.

```
#include <BTypes.h>
```

Public Member Functions

- BDataChunk (void *data=0, BUInt size=0)

Public Attributes

- void * data
- BUInt size

7.14.1 Detailed Description

A chunk of data allowing writes of multiple chunks of segmented data.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 BDataChunk()

```
BDataChunk::BDataChunk (
            void * data = 0,
            BUInt size = 0 )  [inline]
```

7.14.3 Member Data Documentation

7.14.3.1 data

```
void* BDataChunk::data
```

7.14.3.2 size

```
BUInt BDataChunk::size
```

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

- BTypes.h

7.15 BDate Class Reference

This class store a UTC calendar date as a year and a year's day.

`#include <BDate.h>`

Public Member Functions

- BDate (int year=0, int month=1, int day=1)
- BDate (BString str)
- ~BDate ()
- void clear ()

  Clear the date/time.
- void setFirst ()

  Set the first date available.
- void setLast ()

  Set the last date available.
- void set (time_t time)

  Set time using Unix time (seconds from 1970-01-01)
- void set (int year=0, int month=1, int day=1)
- void setYDay (int year=0, int yday=0)
- void setNow ()

  Set the timeStamp to now.
- int year ()
- int yday ()
- int month ()
- int day ()
- void getDate (int &year, int &mon, int &day)
- BString getString ()

  Get the time as an ISO date/time string.
- BString getStringFormatted (BString format)

  Gets the time in a string form as per the format. Format syntax as per strftime()
- BError setString (BString str)

  Set the time from an ISO date/time.
- int isSet ()

  Check if the date has been set.
- int compare (const BDate &date) const

  Compare two dates.
- operator BString ()
- int operator== (const BDate &date) const
- int operator!= (const BDate &date) const
- int operator> (const BDate &date) const
- int operator>= (const BDate &date) const
- int operator< (const BDate &date) const
- int operator<= (const BDate &date) const

Static Public Member Functions

- static int isLeap (int year)
- static int daysInMonth (int year, int month)

Public Attributes

- uint16_t oyear

  Year (0 .. 65535)
- uint16_t oyday

  Day in year (0 .. 365)

7.15.1 Detailed Description

This class store a UTC calendar date as a year and a year's day.

The BDate class stores a calendar date wit a year and day in the year components. It provides functions to set this date from a string and convert the data to a string as well as BDate comparison functions.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 BDate() [1/2]

```
BDate::BDate (
            int year = 0,
            int month = 1,
            int day = 1 )
```

7.15.2.2 BDate() [2/2]

```
BDate::BDate (
            BString str )
```

7.15.2.3 ~BDate()

```
BDate::~BDate ( )
```

7.15.3 Member Function Documentation

7.15.3.1 clear()

```
void BDate::clear ( )
```

Clear the date/time.

7.15.3.2 setFirst()

```
void BDate::setFirst ( )
```

Set the first date available.

7.15.3.3 setLast()

```
void BDate::setLast ( )
```

Set the last date available.

7.15.3.4 set() [1/2]

```
void BDate::set (
            time_t time )
```

Set time using Unix time (seconds from 1970-01-01)

7.15.3.5 set() [2/2]

```
void BDate::set (
            int year = 0,
            int month = 1,
            int day = 1 )
```

7.15.3.6 setYDay()

```
void BDate::setYDay (
            int year = 0,
            int yday = 0 )
```

7.15.3.7 setNow()

```
void BDate::setNow ( )
```

Set the timeStamp to now.

7.15.3.8 year()

```
int BDate::year ( )
```

7.15.3.9 yday()

```
int BDate::yday ( )
```

7.15.3.10 month()

```
int BDate::month ( )
```

7.15.3.11 day()

```
int BDate::day ( )
```

7.15.3.12 getDate()

```
void BDate::getDate (
             int & year,
             int & mon,
             int & day )
```

7.15.3.13 getString()

```
BString BDate::getString ( )
```

Get the time as an ISO date/time string.

7.15.3.14 getStringFormatted()

```
BString BDate::getStringFormatted (
            BString format )
```

Gets the time in a string form as per the format. Format syntax as per strftime()

7.15.3.15 setString()

```
BError BDate::setString (
            BString str )
```

Set the time from an ISO date/time.

7.15.3.16 isSet()

```
int BDate::isSet ( )  [inline]
```

Check if the date has been set.

7.15.3.17 compare()

```
int BDate::compare (
            const BDate & date ) const
```

Compare two dates.

7.15.3.18 operator BString()

```
BDate::operator BString ( )  [inline]
```

7.15.3.19 operator==()

```
int BDate::operator== (
            const BDate & date ) const  [inline]
```

7.15.3.20 operator"!=()

```
int BDate::operator!= (
            const BDate & date ) const  [inline]
```

7.15.3.21 operator>()

```
int BDate::operator> (
            const BDate & date ) const  [inline]
```

7.15.3.22 operator>=()

```
int BDate::operator>= (
            const BDate & date ) const  [inline]
```

7.15.3.23 operator<()

```
int BDate::operator< (
            const BDate & date ) const  [inline]
```

7.15.3.24 operator<=()

```
int BDate::operator<= (
            const BDate & date ) const  [inline]
```

7.15.3.25 isLeap()

```
int BDate::isLeap (
            int year )  [static]
```

7.15.3.26 daysInMonth()

```
int BDate::daysInMonth (
            int year,
            int month )  [static]
```

7.15.4 Member Data Documentation

7.15.4.1 oyear

`uint16_t BDate::oyear`

Year (0 .. 65535)

7.15.4.2 oyday

`uint16_t BDate::oyday`

Day in year (0 .. 365)

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

- BDate.h
- BDate.cpp

7.16 BDebugBacktrace Class Reference

Backtrace on crash class.

`#include <BDebug.h>`

Public Member Functions

- BDebugBacktrace ()
- ~BDebugBacktrace ()
- void dumpBacktraceStdout (char *comment)
- int dumpBacktraceFile (char *fileName, char *comment)
- void dumpBacktraceSyslog (char *comment)
- void dumpBacktrace (char *strBuf, int strBufLen, char *comment)

7.16.1 Detailed Description

Backtrace on crash class.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 BDebugBacktrace()

```
BDebugBacktrace::BDebugBacktrace ( )
```

7.16.2.2 ~BDebugBacktrace()

```
BDebugBacktrace::~BDebugBacktrace ( )
```

7.16.3 Member Function Documentation

7.16.3.1 dumpBacktraceStdout()

```
void BDebugBacktrace::dumpBacktraceStdout (
            char * comment )
```

7.16.3.2 dumpBacktraceFile()

```
int BDebugBacktrace::dumpBacktraceFile (
            char * fileName,
            char * comment )
```

7.16.3.3 dumpBacktraceSyslog()

```
void BDebugBacktrace::dumpBacktraceSyslog (
            char * comment )
```

7.16.3.4 dumpBacktrace()

```
void BDebugBacktrace::dumpBacktrace (
            char * strBuf,
            int strBufLen,
            char * comment )
```

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

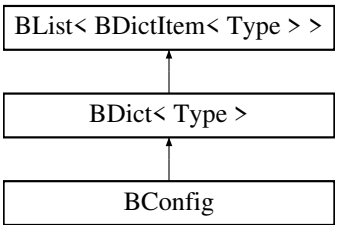
- BDebug.h

7.17 BDict< Type > Class Template Reference

Dictionary list class using templates.

`#include <BDict.h>`

Inheritance diagram for BDict< Type >:



Public Types

- typedef BIter iterator

Public Member Functions

- BDict (int hashSize=100)
- BDict (const BDict< Type > &dict)
- int hasKey (const BString &k) const
- BString key (const BIter &i) const
- void clear ()

  Clear the list.
- void insert (BIter &i, const BDictItem< Type > &item)
- void append (const BDictItem< Type > &item)
- void append (const BDict< Type > &dict)
- void del (const BString &k)
- void del (BIter &i)

  Delete specified item.
- BIter find (const BString &k) const
- Type & operator[ ] (const BString &i)
- const Type & operator[ ] (const BString &i) const
- Type & operator[ ] (const BIter &i)
- const Type & operator[ ] (const BIter &i) const
- Type & operator[ ] (int i)
- const Type & operator[ ] (int i) const
- BDict< Type > operator+ (const BDict< Type > &dict) const
- BDict< Type > & operator= (const BDict< Type > &dict)
- void hashPrint ()

Additional Inherited Members

7.17.1 Detailed Description

template<class Type>
class BDict< Type >

Dictionary list class using templates.

7.17.2 Member Typedef Documentation

7.17.2.1 iterator

```
template<class Type >
typedef BIter BDict< Type >::iterator
```

7.17.3 Constructor & Destructor Documentation

7.17.3.1 BDict() [1/2]

```
template<class Type >
BDict< Type >::BDict (
            int hashSize = 100 )
```

7.17.3.2 BDict() [2/2]

```
template<class Type >
BDict< Type >::BDict (
            const BDict< Type > & dict )
```

7.17.4 Member Function Documentation

7.17.4.1 hasKey()

```
template<class Type >
int BDict< Type >::hasKey (
            const BString & k ) const
```

7.17.4.2 key()

```
template<class Type >
BString BDict< Type >::key (
            const BIter & i ) const
```

7.17.4.3 clear()

```
template<class Type >
void BDict< Type >::clear  [virtual]
```

Clear the list.

Reimplemented from BList< BDictItem< Type > >.

7.17.4.4 insert()

```
template<class Type >
void BDict< Type >::insert (
            BIter & i,
            const BDictItem< Type > & item )
```

7.17.4.5 append() [1/2]

```
template<class Type >
void BDict< Type >::append (
            const BDictItem< Type > & item )
```

7.17.4.6 append() [2/2]

```
template<class Type >
void BDict< Type >::append (
            const BDict< Type > & dict )
```

7.17.4.7 del() [1/2]

```
template<class Type >
void BDict< Type >::del (
            const BString & k )
```

7.17.4.8 del() [2/2]

```
template<class Type >
void BDict< Type >::del (
            BIter & i )  [virtual]
```

Delete specified item.

Reimplemented from BList< BDictItem< Type > >.

7.17.4.9 find()

```
template<class Type >
BIter BDict< Type >::find (
            const BString & k ) const
```

7.17.4.10 operator[]() [1/6]

```
template<class Type >
Type & BDict< Type >::operator[] (
            const BString & i )
```

7.17.4.11 operator[]() [2/6]

```
template<class Type >
const Type & BDict< Type >::operator[] (
            const BString & i ) const
```

7.17.4.12 operator[]() [3/6]

```
template<class Type >
Type & BDict< Type >::operator[] (
            const BIter & i )
```

7.17.4.13 operator[]() [4/6]

```
template<class Type >
const Type & BDict< Type >::operator[] (
            const BIter & i ) const
```

7.17.4.14 operator[]() [5/6]

```
template<class Type >
Type & BDict< Type >::operator[] (
            int i )
```

7.17.4.15 operator[]() [6/6]

```
template<class Type >
const Type & BDict< Type >::operator[] (
            int i ) const
```

7.17.4.16 operator+()

```
template<class Type >
BDict< Type > BDict< Type >::operator+ (
            const BDict< Type > & dict ) const
```

7.17.4.17 operator=()

```
template<class Type >
BDict< Type > & BDict< Type >::operator= (
            const BDict< Type > & dict )
```

7.17.4.18 hashPrint()

```
template<class Type >
void BDict< Type >::hashPrint
```

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

- BDict.h

7.18 BDictItem< Type > Class Template Reference

Template based Dictionary classes item.

```
#include <BDict.h>
```

Public Member Functions

- BDictItem (BString k="", Type v=Type())

Public Attributes

- BString key
- Type value

7.18.1 Detailed Description

template<class Type>
class BDictItem< Type >

Template based Dictionary classes item.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 BDictItem()

```
template<class Type >
BDictItem< Type >::BDictItem (
            BString k = "",
            Type v = Type() )  [inline]
```

7.18.3 Member Data Documentation

7.18.3.1 key

```
template<class Type >
BString BDictItem< Type >::key
```

7.18.3.2 value

```
template<class Type >
Type BDictItem< Type >::value
```

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

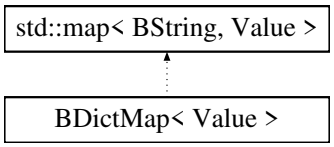
- BDict.h

7.19 BDictMap< Value > Class Template Reference

Mapped Dictionary class.

`#include <BDictMap.h>`

Inheritance diagram for BDictMap< Value >:



Public Types

- typedef BDictMap< Value >::iterator iterator

Public Member Functions

- void clear ()
- int hasKey (const BString &k)
- BString key (iterator &i)
- unsigned int size ()
- void start (iterator &i)
- int isEnd (iterator &i)
- void next (iterator &i)
- void del (const iterator &i)
- void del (const BString &k)
- Value & operator[ ] (iterator &i)
- Value & operator[ ] (const BString &i)

7.19.1 Detailed Description

template<typename Value>
class BDictMap< Value >

Mapped Dictionary class.

This is based on the Standard C++ library map class and has all of the functionality of that class.

7.19.2 Member Typedef Documentation

7.19.2.1 iterator

```
template<typename Value >
typedef BDictMap<Value>::iterator BDictMap< Value >::iterator
```

7.19.3 Member Function Documentation

7.19.3.1 clear()

```
template<typename Value >
void BDictMap< Value >::clear ( )  [inline]
```

7.19.3.2 hasKey()

```
template<typename Value >
int BDictMap< Value >::hasKey (
            const BString & k )  [inline]
```

7.19.3.3 key()

```
template<typename Value >
BString BDictMap< Value >::key (
            iterator & i )  [inline]
```

7.19.3.4 size()

```
template<typename Value >
unsigned int BDictMap< Value >::size ( )  [inline]
```

7.19.3.5 start()

```
template<typename Value >
void BDictMap< Value >::start (
            iterator & i )  [inline]
```

7.19.3.6 isEnd()

```
template<typename Value >
int BDictMap< Value >::isEnd (
            iterator & i )  [inline]
```

7.19.3.7 next()

```
template<typename Value >
void BDictMap< Value >::next (
            iterator & i )  [inline]
```

7.19.3.8 del() [1/2]

```
template<typename Value >
void BDictMap< Value >::del (
            const iterator & i )  [inline]
```

7.19.3.9 del() [2/2]

```
template<typename Value >
void BDictMap< Value >::del (
            const BString & k )  [inline]
```

7.19.3.10 operator[]() [1/2]

```
template<typename Value >
Value& BDictMap< Value >::operator[] (
            iterator & i )  [inline]
```

7.19.3.11 operator[]() [2/2]

```
template<typename Value >
Value& BDictMap< Value >::operator[] (
            const BString & i )  [inline]
```

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

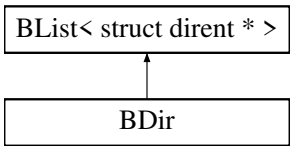
- BDictMap.h

7.20 BDir Class Reference

File system directory class.

`#include <BDir.h>`

Inheritance diagram for BDir:



Public Member Functions

- BDir ()
- BDir (BString name)
- ~BDir ()
- BError open (BString name)

  Reads named directory.
- BError error ()

  Current value of error.
- BError read ()

  read/re-reads directory
- void clear ()

  Clears list.
- void setWild (BString wild)

  Set wildcard filter string used on read.
- void setSort (int on)

  Set alpha sort on/off.
- BString entryName (BIter i)

  Get filename.
- struct stat entryStat (BIter i)

  Get file stats.
- struct stat64 entryStat64 (BIter i)

  Get file stats 64.

Additional Inherited Members

7.20.1 Detailed Description

File system directory class.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 BDir() [1/2]

```
BDir::BDir ( )
```

7.20.2.2 BDir() [2/2]

```
BDir::BDir (
            BString name )
```

7.20.2.3 ~BDir()

```
BDir::~BDir ( )
```

7.20.3 Member Function Documentation

7.20.3.1 open()

```
BError BDir::open (
            BString name )
```

Reads named directory.

7.20.3.2 error()

```
BError BDir::error ( )
```

Current value of error.

7.20.3.3 read()

```
BError BDir::read ( )
```

read/re-reads directory

7.20.3.4 clear()

```
void BDir::clear ( )  [virtual]
```

Clears list.

Reimplemented from BList< struct dirent * >.

7.20.3.5 setWild()

```
void BDir::setWild (
            BString wild )
```

Set wildcard filter string used on read.

7.20.3.6 setSort()

```
void BDir::setSort (
            int on )
```

Set alpha sort on/off.

7.20.3.7 entryName()

```
BString BDir::entryName (
            BIter i )
```

Get filename.

7.20.3.8 entryStat()

```
struct stat BDir::entryStat (
            BIter i )
```

Get file stats.

7.20.3.9 entryStat64()

```
struct stat64 BDir::entryStat64 (
            BIter i )
```

Get file stats 64.

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

- BDir.h
- BDir.cpp

7.21 BDuration Class Reference

Stores and manipulates a time to the nearest microsecond and a maximum of 24 hours.

`#include <BDuration.h>`

Public Member Functions

- BDuration (int hour=0, int minute=0, int second=0, int microsecond=0)
- BDuration (BString str)
- ~BDuration ()
- void clear ()

  Clear the duration.
- void set (int hour=0, int minute=0, int second=0, int microsecond=0)
- void addMilliSeconds (int64_t milliSeconds)

  Add the given number of milli seconds.
- void addMicroSeconds (int64_t microSeconds)

  Add the given number of micro seconds.
- void addSeconds (int seconds)

  Add the given number of seconds.
- uint32_t getSeconds ()

  Get number of seconds.
- uint64_t getMicroSeconds ()

  Get number of micro seconds.
- int hour ()
- int minute ()
- int second ()
- int microSecond ()
- BString getString ()

  Get the time as an ISO date/time string.
- BError setString (BString time)

  Set the time from an ISO date/time.

7.21.1 Detailed Description

Stores and manipulates a time to the nearest microsecond and a maximum of 24 hours.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 BDuration() [1/2]

```
BDuration::BDuration (
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.21.2.2 BDuration() [2/2]

```
BDuration::BDuration (
            BString str )
```

7.21.2.3 ~BDuration()

```
BDuration::~BDuration ( )
```

7.21.3 Member Function Documentation

7.21.3.1 clear()

```
void BDuration::clear ( )
```

Clear the duration.

7.21.3.2 set()

```
void BDuration::set (
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.21.3.3 addMilliSeconds()

```
void BDuration::addMilliSeconds (
            int64_t milliSeconds )
```

Add the given number of milli seconds.

7.21.3.4 addMicroSeconds()

```
void BDuration::addMicroSeconds (
            int64_t microSeconds )
```

Add the given number of micro seconds.

7.21.3.5 addSeconds()

```
void BDuration::addSeconds (
            int seconds )
```

Add the given number of seconds.

7.21.3.6 getSeconds()

```
uint32_t BDuration::getSeconds ( )
```

Get number of seconds.

7.21.3.7 getMicroSeconds()

```
uint64_t BDuration::getMicroSeconds ( )
```

Get number of micro seconds.

7.21.3.8 hour()

```
int BDuration::hour ( )
```

7.21.3.9 minute()

```
int BDuration::minute ( )
```

7.21.3.10 second()

```
int BDuration::second ( )
```

7.21.3.11 microSecond()

```
int BDuration::microSecond ( )
```

7.21.3.12 getString()

```
BString BDuration::getString ( )
```

Get the time as an ISO date/time string.

7.21.3.13 setString()

```
BError BDuration::setString (
            BString time )
```

Set the time from an ISO date/time.

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

- BDuration.h
- BDuration.cpp

7.22 BEntry Class Reference

Manipulate a name value pair.

```
#include <BEntry.h>
```

Public Member Functions

- BEntry ()
- BEntry (BString name, BString value)

  Set name and value.
- BEntry (BString line)

  Set name and value from white space deliminated string.
- BString getName ()

  Get the name.
- BString getValue ()

  Get the value.
- void setLine (BString line)

  Set name and value from white space deliminated string.
- void setName (BString name)

  Set the name.
- void setValue (BString value)

  Set the value.
- BString line ()

  Return name and value as padded single string.
- void print ()

  Print name and value.

7.22.1 Detailed Description

Manipulate a name value pair.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 BEntry() [1/3]

```
BEntry::BEntry ( )
```

7.22.2.2 BEntry() [2/3]

```
BEntry::BEntry (
            BString name,
            BString value )
```

Set name and value.

7.22.2.3 BEntry() [3/3]

```
BEntry::BEntry (
            BString line )
```

Set name and value from white space deliminated string.

7.22.3 Member Function Documentation

7.22.3.1 getName()

```
BString BEntry::getName ( )
```

Get the name.

7.22.3.2 getValue()

```
BString BEntry::getValue ( )
```

Get the value.

7.22.3.3 setLine()

```
void BEntry::setLine (
            BString line )
```

Set name and value from white space deliminated string.

7.22.3.4 setName()

```
void BEntry::setName (
            BString name )
```

Set the name.

7.22.3.5 setValue()

```
void BEntry::setValue (
            BString value )
```

Set the value.

7.22.3.6 line()

```
BString BEntry::line ( )
```

Return name and value as padded single string.

7.22.3.7 print()

```
void BEntry::print ( )
```

Print name and value.

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

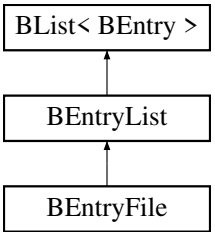
- BEntry.h
- BEntry.cpp

7.23 BEntryFile Class Reference

A file based list of string name/value pairs.

```
#include <BEntry.h>
```

Inheritance diagram for BEntryFile:



Public Member Functions

- BEntryFile ()
- BEntryFile (BString filename)

  Opens entryfile.
- ~BEntryFile ()
- int open (BString filename)

  Opens entryfile.
- int read ()

  Reads entry file and builds list.
- int write ()

  Writes list to entryfile.
- int writeList (BEntryList &l)

  Writes specified list to file.
- void clear ()

  Clears current list.
- BString filename ()

  Returns the filename.

Additional Inherited Members

7.23.1 Detailed Description

A file based list of string name/value pairs.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 BEntryFile() [1/2]

```
BEntryFile::BEntryFile ( )
```

7.23.2.2 BEntryFile() [2/2]

```
BEntryFile::BEntryFile (
            BString filename )
```

Opens entryfile.

7.23.2.3 ~BEntryFile()

```
BEntryFile::~BEntryFile ( )
```

7.23.3 Member Function Documentation

7.23.3.1 open()

```
int BEntryFile::open (
            BString filename )
```

Opens entryfile.

7.23.3.2 read()

```
int BEntryFile::read ( )
```

Reads entry file and builds list.

7.23.3.3 write()

```
int BEntryFile::write ( )
```

Writes list to entryfile.

7.23.3.4 writeList()

```
int BEntryFile::writeList (
            BEntryList & l )
```

Writes specified list to file.

7.23.3.5 clear()

```
void BEntryFile::clear ( )  [virtual]
```

Clears current list.

Reimplemented from BEntryList.

7.23.3.6 filename()

```
BString BEntryFile::filename ( )
```

Returns the filename.

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

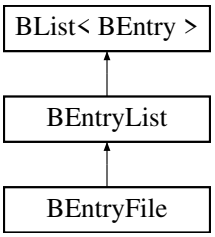
- BEntry.h
- BEntry.cpp

7.24 BEntryList Class Reference

List of Entries. Where each entry is a name value pair.

`#include <BEntry.h>`

Inheritance diagram for BEntryList:



Public Member Functions

- BEntryList ()
- int isSet (BString name)

  1 if name is in list and value is set
- BEntry * find (BString name)

  Returns entry if name is found otherwise NULL.
- BString findValue (BString name)

  Returns value of name. Returns "" if name not found.
- int setValue (BString name, BString value)

  Set the value of name. Returns 0 if name not found.
- int setValueRaw (BString name, BString value)

  Raw setting of value without looking up existing entry.
- void deleteEntry (BString name)

  Deletes the entry.
- void print ()

  Print list.
- BString getString ()

  Return list as string. Each Entry padded and on a new line.
- void insert (BIter &i, const BEntry &item)
- void del (BIter &i)

  Delete specified item.
- void clear ()

  Clear the list.
- BEntryList & operator= (const BEntryList &l)

Additional Inherited Members

7.24.1 Detailed Description

List of Entries. Where each entry is a name value pair.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 BEntryList()

```
BEntryList::BEntryList ( )
```

7.24.3 Member Function Documentation

7.24.3.1 isSet()

```
int BEntryList::isSet (
            BString name )
```

1 if name is in list and value is set

7.24.3.2 find()

```
BEntry * BEntryList::find (
            BString name )
```

Returns entry if name is found otherwise NULL.

7.24.3.3 findValue()

```
BString BEntryList::findValue (
            BString name )
```

Returns value of name. Returns "" if name not found.

7.24.3.4 setValue()

```
int BEntryList::setValue (
            BString name,
            BString value )
```

Set the value of name. Returns 0 if name not found.

7.24.3.5 setValueRaw()

```
int BEntryList::setValueRaw (
            BString name,
            BString value )
```

Raw setting of value without looking up existing entry.

7.24.3.6 deleteEntry()

```
void BEntryList::deleteEntry (
            BString name )
```

Deletes the entry.

7.24.3.7 print()

```
void BEntryList::print ( )
```

Print list.

7.24.3.8 getString()

```
BString BEntryList::getString ( )
```

Return list as string. Each Entry padded and on a new line.

7.24.3.9 insert()

```
void BEntryList::insert (
            BIter & i,
            const BEntry & item )
```

7.24.3.10 del()

```
void BEntryList::del (
            BIter & i )  [virtual]
```

Delete specified item.

Reimplemented from BList< BEntry >.

7.24.3.11 clear()

```
void BEntryList::clear ( )  [virtual]
```

Clear the list.

Reimplemented from BList< BEntry >.

Reimplemented in BEntryFile.

7.24.3.12 operator=()

```
BEntryList & BEntryList::operator= (
            const BEntryList & l )
```

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

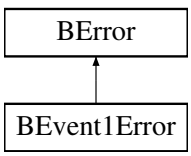
- BEntry.h
- BEntry.cpp

7.25 BError Class Reference

Error return class. This class is used to return the error status from a function. It encapsulates an integer error number and a string.

```
#include <BError.h>
```

Inheritance diagram for BError:



Public Member Functions

- BError (int errNo=ErrorOk, BString errStr="")
  Create object.
- BError (BString errStr)
  Create with error set and error string.
- BError copy ()
  Return an independant copy.
- BError & set (int errNo, BString errStr="")
  Set error number and message.
- BError & clear ()
  Clear the error.
- BError & setError (BString errStr="")
  Set error type ERROR with optional message.
- BString getString () const
  Get error message.
- int getNumber () const
  Get The error number.
- int num () const
  Get The error number.
- const char * str () const
  Return a char string.*
- int getErrorNo () const
  Get The error number.
- operator int () const
  Return error number.

7.25.1 Detailed Description

Error return class. This class is used to return the error status from a function. It encapsulates an integer error number and a string.

An error number of ErrorOk (0) indicates no error, a value of ErrorMisc (1) indicates some error and a value of ErrorWarning (2) indiactes a warning. Specific error numbers are defeined in BErrorNum. System low level errors (errno) are defined by negativer values. Specific application errors are those above the value 64.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 BError() [1/2]

```
BError::BError (
            int errNo = ErrorOk,
            BString errStr = "" )
```

Create object.

7.25.2.2 BError() [2/2]

```
BError::BError (
            BString errStr )
```

Create with error set and error string.

7.25.3 Member Function Documentation

7.25.3.1 copy()

```
BError BError::copy ( )
```

Return an independant copy.

7.25.3.2 set()

```
BError & BError::set (
            int errNo,
            BString errStr = "" )
```

Set error number and message.

7.25.3.3 clear()

```
BError & BError::clear ( )
```

Clear the error.

7.25.3.4 setError()

```
BError & BError::setError (
            BString errStr = "" )
```

Set error type ERROR with optional message.

7.25.3.5 getString()

```
BString BError::getString ( ) const
```

Get error message.

7.25.3.6 getNumber()

```
int BError::getNumber ( ) const
```

Get The error number.

7.25.3.7 num()

```
int BError::num ( ) const
```

Get The error number.

7.25.3.8 str()

```
const char * BError::str ( ) const
```

Return a char* string.

7.25.3.9 getErrorNo()

```
int BError::getErrorNo ( ) const
```

Get The error number.

7.25.3.10 operator int()

```
BError::operator int ( ) const  [inline]
```

Return error number.

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

- BError.h
- BError.cpp

7.26 BErrorTime Class Reference

Error return class with time field.

`#include <BErrorTime.h>`

Public Types

- enum Type { None = 0 , Error = 1 }

Public Member Functions

- BErrorTime (int errNo=None, BTimeStamp errTime=BTimeStamp(), BString errStr="")
  Create object.
- BErrorTime & set (int errNo, BTimeStamp errTime=BTimeStamp(), BString errStr="")
  Set error number and message.
- BErrorTime & clear ()
  Clear the error.
- int getErrorNo () const
  Get The error number.
- BTimeStamp getTime () const
  Get time.
- BString getString () const
  Get error message.
- BErrorTime copy ()
  Return an independant copy.
- operator int () const
  Return error number.

7.26.1 Detailed Description

Error return class with time field.

This provides an alternative to the standard BError return type for errors but includes BTimeStamp information.

7.26.2 Member Enumeration Documentation

7.26.2.1 Type

`enum BErrorTime::Type`

Enumerator

None	
Error	

7.26.3 Constructor & Destructor Documentation

7.26.3.1 BErrorTime()

```
BErrorTime::BErrorTime (
            int errNo = None,
            BTimeStamp errTime = BTimeStamp(),
            BString errStr = "" )
```

Create object.

7.26.4 Member Function Documentation

7.26.4.1 set()

```
BErrorTime & BErrorTime::set (
            int errNo,
            BTimeStamp errTime = BTimeStamp(),
            BString errStr = "" )
```

Set error number and message.

7.26.4.2 clear()

```
BErrorTime & BErrorTime::clear ( )
```

Clear the error.

7.26.4.3 getErrorNo()

```
int BErrorTime::getErrorNo ( ) const
```

Get The error number.

7.26.4.4 getTime()

```
BTimeStamp BErrorTime::getTime ( ) const
```

Get time.

7.26.4.5 getString()

```
BString BErrorTime::getString ( ) const
```

Get error message.

7.26.4.6 copy()

```
BErrorTime BErrorTime::copy ( )
```

Return an independant copy.

7.26.4.7 operator int()

```
BErrorTime::operator int ( ) const
```

Return error number.

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

- BErrorTime.h
- BErrorTime.cpp

7.27 BEvent Class Reference

An event description class.

```
#include <BEvent.h>
```

Public Member Functions

- BEvent (BUInt32 type=BEventTypeNone, BUInt32 arg=0)
- BUInt32 type ()
- BUInt32 arg ()

7.27.1 Detailed Description

An event description class.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 BEvent()

```
BEvent::BEvent (
            BUInt32 type = BEventTypeNone,
            BUInt32 arg = 0 )
```

7.27.3 Member Function Documentation

7.27.3.1 type()

```
BUInt32 BEvent::type ( )
```

7.27.3.2 arg()

```
BUInt32 BEvent::arg ( )
```

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

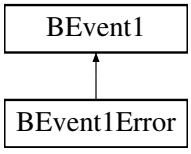
- BEvent.h
- BEvent.cpp

7.28 BEvent1 Class Reference

This class provides a base class for all event objects that can be sent over the events interface.

```
#include <BEvent1.h>
```

Inheritance diagram for BEvent1:



Public Member Functions

- BEvent1 (uint32_t type)
- virtual ~BEvent1 ()
- uint32_t getType ()
- virtual BError getBinary (void *data, uint32_t &size)
- virtual BError setBinary (void *data, uint32_t &size)

7.28.1 Detailed Description

This class provides a base class for all event objects that can be sent over the events interface.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 BEvent1()

```
BEvent1::BEvent1 (
            uint32_t type )
```

7.28.2.2 ~BEvent1()

```
BEvent1::~BEvent1 ( )  [virtual]
```

7.28.3 Member Function Documentation

7.28.3.1 getType()

```
uint32_t BEvent1::getType ( )
```

7.28.3.2 getBinary()

```
BError BEvent1::getBinary (
            void * data,
            uint32_t & size )  [virtual]
```

Reimplemented in BEvent1Error.

7.28.3.3 setBinary()

```
BError BEvent1::setBinary (
            void * data,
            uint32_t & size )  [virtual]
```

Reimplemented in BEvent1Error.

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

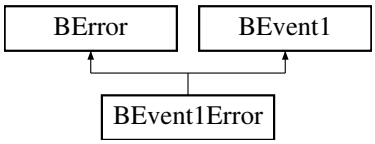
- BEvent1.h
- BEvent1.cpp

7.29 BEvent1Error Class Reference

This class provides a class to send a BError event.

```
#include <BEvent1.h>
```

Inheritance diagram for BEvent1Error:



Public Member Functions

- BEvent1Error (int errNo=ErrorOk, BString errStr="")
- BError getBinary (void *data, uint32_t &size)
- BError setBinary (void *data, uint32_t &size)

7.29.1 Detailed Description

This class provides a class to send a BError event.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 BEvent1Error()

```
BEvent1Error::BEvent1Error (
            int errNo = ErrorOk,
            BString errStr = "" )
```

7.29.3 Member Function Documentation

7.29.3.1 getBinary()

```
BError BEvent1Error::getBinary (
            void * data,
            uint32_t & size )  [virtual]
```

Reimplemented from BEvent1.

7.29.3.2 setBinary()

```
BError BEvent1Error::setBinary (
            void * data,
            uint32_t & size )  [virtual]
```

Reimplemented from BEvent1.

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

- BEvent1.h
- BEvent1.cpp

7.30 BEvent1Int Class Reference

This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

```
#include <BEvent1.h>
```

Public Member Functions

- BEvent1Int ()
- ~BEvent1Int ()
- void clear ()

  Clear events pending.
- BError sendEvent (int event)

  Send an event.
- BError getEvent (int &event, int timeOutUs=-1)

  Receive the event.
- int getFd ()

7.30.1 Detailed Description

This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 BEvent1Int()

```
BEvent1Int::BEvent1Int ( )
```

7.30.2.2 ~BEvent1Int()

```
BEvent1Int::~BEvent1Int ( )
```

7.30.3 Member Function Documentation

7.30.3.1 clear()

```
void BEvent1Int::clear ( )
```

Clear events pending.

7.30.3.2 sendEvent()

```
BError BEvent1Int::sendEvent (
            int event )
```

Send an event.

7.30.3.3 getEvent()

```
BError BEvent1Int::getEvent (
            int & event,
            int timeOutUs = -1 )
```

Receive the event.

7.30.3.4 getFd()

```
int BEvent1Int::getFd ( )
```

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

- BEvent1.h
- BEvent1.cpp

7.31 BEvent1Pipe Class Reference

This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.

```
#include <BEvent1.h>
```

Public Member Functions

- BEvent1Pipe ()
- ~BEvent1Pipe ()
- void clear ()

  Clear events pending.
- BError sendEvent (BEvent1 *event)

  Send an event.
- BError getEvent (BEvent1 *event, int timeOutUs=-1)

  Receive the event.
- int getReceiveFd ()

  returns the receive file descriptor for the poll system call

7.31.1 Detailed Description

This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 BEvent1Pipe()

```
BEvent1Pipe::BEvent1Pipe ( )
```

7.31.2.2 ~BEvent1Pipe()

```
BEvent1Pipe::~BEvent1Pipe ( )
```

7.31.3 Member Function Documentation

7.31.3.1 clear()

```
void BEvent1Pipe::clear ( )
```

Clear events pending.

7.31.3.2 sendEvent()

```
BError BEvent1Pipe::sendEvent (
            BEvent1 * event )
```

Send an event.

7.31.3.3 getEvent()

```
BError BEvent1Pipe::getEvent (
            BEvent1 * event,
            int timeOutUs = -1 )
```

Receive the event.

7.31.3.4 getReceiveFd()

```
int BEvent1Pipe::getReceiveFd ( )
```

returns the receive file descriptor for the poll system call

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

- BEvent1.h
- BEvent1.cpp

7.32 BEventPipe Class Reference

This class provides an interface for sending simple integer events via a pipe file descriptor.

`#include <BEvent.h>`

Public Member Functions

- BEventPipe ()
- ~BEventPipe ()
- void clear ()

  Clear events pending.
- int getFd ()
- BUInt writeAvailable () const
- BError write (const BEvent &event, BTimeout timeout=BTimeoutForever)

  Append an item onto the queue.
- BUInt readAvailable () const
- BError read (BEvent &event, BTimeout timeout=BTimeoutForever)

  Get an item from the queue.

7.32.1 Detailed Description

This class provides an interface for sending simple integer events via a pipe file descriptor.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 BEventPipe()

`BEventPipe::BEventPipe ( )`

7.32.2.2 ~BEventPipe()

`BEventPipe::~BEventPipe ( )`

7.32.3 Member Function Documentation

7.32.3.1 clear()

```
void BEventPipe::clear ( )
```

Clear events pending.

7.32.3.2 getFd()

```
int BEventPipe::getFd ( )
```

7.32.3.3 writeAvailable()

```
BUInt BEventPipe::writeAvailable ( ) const
```

7.32.3.4 write()

```
BError BEventPipe::write (
            const BEvent & event,
            BTimeout timeout = BTimeoutForever )
```

Append an item onto the queue.

7.32.3.5 readAvailable()

```
BUInt BEventPipe::readAvailable ( ) const
```

7.32.3.6 read()

```
BError BEventPipe::read (
            BEvent & event,
            BTimeout timeout = BTimeoutForever )
```

Get an item from the queue.

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

- BEvent.h
- BEvent.cpp

7.33 BFifo< Type > Class Template Reference

A template first in first out data buffer to store any object types.

`#include <BFifo.h>`

Public Member Functions

- BFifo (BUInt size)
- ~BFifo ()
- void clear ()
- BUInt size ()

  Returns fifo size.
- BError resize (BUInt size)

  Resize FIFO, clears it as well.
- BError rebase ()

  Rebases fifo so read pointer is at zero moving memory as needed.
- BUInt writeAvailable ()

  How many items that can be written.
- BUInt writeAvailableChunk ()

  How many items that can be written in a chunk.
- BError write (const Type v)

  Write a single item.
- BError write (const Type *data, BUInt num)

  Write a set of items. Can only write a maximum of writeAvailableChunk() to save going beyond end of FIFO buffer.
- Type * writeData ()

  Returns a pointer to the data.
- Type * writeData (BUInt &num)

  Returns a pointer to the data and how many can be written in a chunk.
- void writeDone (BUInt num)

  Indicates when write is complete.
- void writeBackup (BUInt num)

  Backup, remove num items at end of fifo. Careful, make sure read is not already happening.
- BUInt readAvailable ()

  How many items are available to read.
- BUInt readAvailableChunk ()

  How many items are available to read in a chunk.
- Type read ()

  Read one item.
- BError read (Type *data, BUInt num)

  Read a set of items.
- Type * readData ()

  Returns a pointer to the data.
- Type * readData (BUInt &num)

  Returns a pointer to the data and how many can be read in a chunk.
- void readDone (BUInt num)
- Type readPos (BUInt pos)

  Read item at given offset from current read position.
- void writePos (BUInt pos, const Type &v)

  Write item at given offset from current read position.
- Type & operator[ ] (int pos)

  Direct access to read samples in buffer.

Protected Attributes

- BUInt osize

  The size of the FIFO.
- Type * odata

  FIFO memory buffer.
- volatile BUInt owritePos

  The write pointer.
- volatile BUInt oreadPos

  The read pointer.

7.33.1 Detailed Description

template<class Type>
class BFifo< Type >

A template first in first out data buffer to store any object types.

This class stores data in a ring buffer. It can store up to (size - 1) items. It has separate read and write pointers. These are thread safe. Note the read and write routines do not bounds check the Fifo. You have to use readAvailable()/writeAvailable() functions to check how much can be read/written before performing the reads or writes.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 BFifo()

```
template<class Type >
BFifo< Type >::BFifo (
            BUInt size )
```

7.33.2.2 ~BFifo()

```
template<class Type >
BFifo< Type >::~BFifo ( )
```

7.33.3 Member Function Documentation

7.33.3.1 clear()

```
template<class Type >
void BFifo< Type >::clear ( )
```

7.33.3.2 size()

```
template<class Type >
BUInt BFifo< Type >::size ( )
```

Returns fifo size.

7.33.3.3 resize()

```
template<class Type >
BError BFifo< Type >::resize (
            BUInt size )
```

Resize FIFO, clears it as well.

7.33.3.4 rebase()

```
template<class Type >
BError BFifo< Type >::rebase ( )
```

Rebases fifo so read pointer is at zero moving memory as needed.

7.33.3.5 writeAvailable()

```
template<class Type >
BUInt BFifo< Type >::writeAvailable ( )
```

How many items that can be written.

7.33.3.6 writeAvailableChunk()

```
template<class Type >
BUInt BFifo< Type >::writeAvailableChunk ( )
```

How many items that can be written in a chunk.

7.33.3.7 write() [1/2]

```
template<class Type >
BError BFifo< Type >::write (
            const Type v )
```

Write a single item.

7.33.3.8 write() [2/2]

```
template<class Type >
BError BFifo< Type >::write (
            const Type * data,
            BUInt num )
```

Write a set of items. Can only write a maximum of writeAvailableChunk() to save going beyond end of FIFO buffer.

7.33.3.9 writeData() [1/2]

```
template<class Type >
Type* BFifo< Type >::writeData ( )
```

Returns a pointer to the data.

7.33.3.10 writeData() [2/2]

```
template<class Type >
Type* BFifo< Type >::writeData (
            BUInt & num )
```

Returns a pointer to the data and how many can be written in a chunk.

7.33.3.11 writeDone()

```
template<class Type >
void BFifo< Type >::writeDone (
            BUInt num )
```

Indicates when write is complete.

7.33.3.12 writeBackup()

```
template<class Type >
void BFifo< Type >::writeBackup (
            BUInt num )
```

Backup, remove num items at end of fifo. Careful, make sure read is not already happening.

7.33.3.13 readAvailable()

```
template<class Type >
BUInt BFifo< Type >::readAvailable ( )
```

How many items are available to read.

7.33.3.14 readAvailableChunk()

```
template<class Type >
BUInt BFifo< Type >::readAvailableChunk ( )
```

How many items are available to read in a chunk.

7.33.3.15 read() [1/2]

```
template<class Type >
Type BFifo< Type >::read ( )
```

Read one item.

7.33.3.16 read() [2/2]

```
template<class Type >
BError BFifo< Type >::read (
            Type * data,
            BUInt num )
```

Read a set of items.

7.33.3.17 readData() [1/2]

```
template<class Type >
Type* BFifo< Type >::readData ( )
```

Returns a pointer to the data.

7.33.3.18 readData() [2/2]

```
template<class Type >
Type* BFifo< Type >::readData (
            BUInt & num )
```

Returns a pointer to the data and how many can be read in a chunk.

7.33.3.19 readDone()

```
template<class Type >
void BFifo< Type >::readDone (
            BUInt num )
```

7.33.3.20 readPos()

```
template<class Type >
Type BFifo< Type >::readPos (
            BUInt pos )
```

Read item at given offset from current read position.

7.33.3.21 writePos()

```
template<class Type >
void BFifo< Type >::writePos (
            BUInt pos,
            const Type & v )
```

Write item at given offset from current read position.

7.33.3.22 operator[]()

```
template<class Type >
Type& BFifo< Type >::operator[] (
            int pos )
```

Direct access to read samples in buffer.

7.33.4 Member Data Documentation

7.33.4.1 osize

```
template<class Type >
BUInt BFifo< Type >::osize  [protected]
```

The size of the FIFO.

7.33.4.2 odata

```
template<class Type >
Type* BFifo< Type >::odata  [protected]
```

FIFO memory buffer.

7.33.4.3 owritePos

```
template<class Type >
volatile BUInt BFifo< Type >::owritePos  [protected]
```

The write pointer.

7.33.4.4 oreadPos

```
template<class Type >
volatile BUInt BFifo< Type >::oreadPos  [protected]
```

The read pointer.

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

- BFifo.h

7.34 BFifoCirc< Type > Class Template Reference

This class implements a thread safe FIFO buffer using a binary sized circular memory.

`#include <BFifoCirc.h>`

Public Types

- enum { defaultSize = 1024 }

Public Member Functions

- BFifoCirc (uint32_t size=defaultSize)
- ~BFifoCirc ()
- uint32_t size ()

  Return the buffers actual size.
- void clear ()

  Clear all of the data in the buffer.
- uint32_t writeAvailable ()

  Returns the space available to write.
- BError writeWaitAvailable (uint32_t numFifoSamples)

  Wait for the given number of samples.
- BError write (const Type *data, uint32_t numFifoSamples)

  Writes the data to the buffer. Blocks until complete.
- Type * writeData ()

  Return a pointer to the current start of the buffer.
- void writeDone (uint32_t numFifoSamples)

  Update the write pointer.
- uint32_t readAvailable ()

  Returns the number of bytes of data available.
- BError readWaitAvailable (uint32_t numFifoSamples)

  Wait for given number of samples.
- BError read (Type *data, uint32_t numFifoSamples)
- Type * readData ()

  Pointer to raw data.
- BError readDone (uint32_t numFifoSamples)

  Updates read pointer.
- Type & operator[ ] (int pos)

  Direct access to read samples in buffer.

Protected Member Functions

- BError mapCircularBuffer (uint32_t size)
- void unmapCircularBuffer ()

Protected Attributes

- BMutex olock
- uint32_t ovmSize
- uint32_t osize
- Type * odata
- BFifoCircPos owritePos

  Current write position.
- BCondValue owriteNumFifoSamples

  The number of samples in the FIFO.
- BFifoCircPos oreadPos

  Current read position.

7.34.1 Detailed Description

template<class Type>
class BFifoCirc< Type >

This class implements a thread safe FIFO buffer using a binary sized circular memory.

This class stores data in a ring buffer where the actual buffer is circular. It can store up to (size - 1) items. It has separate read and write pointers. These are thread safe. Note the read and write routines do not bounds check the Fifo. You have to use readAvailable()/writeAvailable() functions to check how much can be read/written before performing the reads or writes. The CPU's MMU is used to create a binary sized memory area that wrapps around on itself.

7.34.2 Member Enumeration Documentation

7.34.2.1 anonymous enum

```
template<class Type >
anonymous enum
```

Enumerator

defaultSize	

7.34.3 Constructor & Destructor Documentation

7.34.3.1 BFifoCirc()

```
template<class Type >
BFifoCirc< Type >::BFifoCirc (
            uint32_t size = defaultSize )
```

7.34.3.2 ~BFifoCirc()

```
template<class Type >
BFifoCirc< Type >::~BFifoCirc ( )
```

7.34.4 Member Function Documentation

7.34.4.1 size()

```
template<class Type >
uint32_t BFifoCirc< Type >::size ( )
```

Return the buffers actual size.

7.34.4.2 clear()

```
template<class Type >
void BFifoCirc< Type >::clear ( )
```

Clear all of the data in the buffer.

7.34.4.3 writeAvailable()

```
template<class Type >
uint32_t BFifoCirc< Type >::writeAvailable ( )
```

Returns the space available to write.

7.34.4.4 writeWaitAvailable()

```
template<class Type >
BError BFifoCirc< Type >::writeWaitAvailable (
            uint32_t numFifoSamples )
```

Wait for the given number of samples.

7.34.4.5 write()

```
template<class Type >
BError BFifoCirc< Type >::write (
            const Type * data,
            uint32_t numFifoSamples )
```

Writes the data to the buffer. Blocks until complete.

7.34.4.6 writeData()

```
template<class Type >
Type* BFifoCirc< Type >::writeData ( )
```

Return a pointer to the current start of the buffer.

7.34.4.7 writeDone()

```
template<class Type >
void BFifoCirc< Type >::writeDone (
            uint32_t numFifoSamples )
```

Update the write pointer.

7.34.4.8 readAvailable()

```
template<class Type >
uint32_t BFifoCirc< Type >::readAvailable ( )
```

Returns the number of bytes of data available.

7.34.4.9 readWaitAvailable()

```
template<class Type >
BError BFifoCirc< Type >::readWaitAvailable (
            uint32_t numFifoSamples )
```

Wait for given number of samples.

7.34.4.10 read()

```
template<class Type >
BError BFifoCirc< Type >::read (
            Type * data,
            uint32_t numFifoSamples )
```

7.34.4.11 readData()

```
template<class Type >
Type* BFifoCirc< Type >::readData ( )
```

Pointer to raw data.

7.34.4.12 readDone()

```
template<class Type >
BError BFifoCirc< Type >::readDone (
            uint32_t numFifoSamples )
```

Updates read pointer.

7.34.4.13 operator[]()

```
template<class Type >
Type& BFifoCirc< Type >::operator[] (
            int pos )
```

Direct access to read samples in buffer.

7.34.4.14 mapCircularBuffer()

```
template<class Type >
BError BFifoCirc< Type >::mapCircularBuffer (
            uint32_t size )  [protected]
```

7.34.4.15 unmapCircularBuffer()

```
template<class Type >
void BFifoCirc< Type >::unmapCircularBuffer ( )  [protected]
```

7.34.5 Member Data Documentation

7.34.5.1 olock

```
template<class Type >
BMutex BFifoCirc< Type >::olock  [protected]
```

7.34.5.2 ovmSize

```
template<class Type >
uint32_t BFifoCirc< Type >::ovmSize  [protected]
```

7.34.5.3 osize

```
template<class Type >
uint32_t BFifoCirc< Type >::osize  [protected]
```

7.34.5.4 odata

```
template<class Type >
Type* BFifoCirc< Type >::odata  [protected]
```

7.34.5.5 owritePos

```
template<class Type >
BFifoCircPos BFifoCirc< Type >::owritePos  [protected]
```

Current write position.

7.34.5.6 owriteNumFifoSamples

```
template<class Type >
BCondValue BFifoCirc< Type >::owriteNumFifoSamples  [protected]
```

The number of samples in the FIFO.

7.34.5.7 oreadPos

```
template<class Type >
BFifoCircPos BFifoCirc< Type >::oreadPos  [protected]
```

Current read position.

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

- BFifoCirc.h

7.35 BFifoCircPos Class Reference

This class implements a pointer into the Fifo's circular buffer.

```
#include <BFifoCirc.h>
```

Public Member Functions

- BFifoCircPos (uint32_t size)
- void setSize (uint32_t size)
- void set (uint32_t pos)

  Sets the position.
- uint32_t pos ()

  The current position.
- void increment (uint32_t numFifoSamples)

  Increment the pointer by the given value.
- uint32_t difference (const BFifoCircPos &pos)

  Return the difference between the two pointers.
- operator int ()
- void operator+= (uint32_t numFifoSamples)
- int operator== (const BFifoCircPos &pos)
- int operator!= (const BFifoCircPos &pos)

7.35.1 Detailed Description

This class implements a pointer into the Fifo's circular buffer.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 BFifoCircPos()

```
BFifoCircPos::BFifoCircPos (
            uint32_t size )
```

7.35.3 Member Function Documentation

7.35.3.1 setSize()

```
void BFifoCircPos::setSize (
            uint32_t size )
```

7.35.3.2 set()

```
void BFifoCircPos::set (
            uint32_t pos )
```

Sets the position.

7.35.3.3 pos()

```
uint32_t BFifoCircPos::pos ( )
```

The current position.

7.35.3.4 increment()

```
void BFifoCircPos::increment (
            uint32_t numFifoSamples )
```

Increment the pointer by the given value.

7.35.3.5 difference()

```
uint32_t BFifoCircPos::difference (
            const BFifoCircPos & pos )
```

Return the difference between the two pointers.

7.35.3.6 operator int()

```
BFifoCircPos::operator int ( )
```

7.35.3.7 operator+=()

```
void BFifoCircPos::operator+= (
            uint32_t numFifoSamples )
```

7.35.3.8 operator==()

```
int BFifoCircPos::operator== (
            const BFifoCircPos & pos )
```

7.35.3.9 operator"!=()

```
int BFifoCircPos::operator!= (
            const BFifoCircPos & pos )
```

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

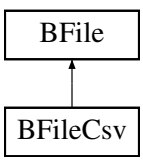
- BFifoCirc.h
- BFifoCirc.cpp

7.36 BFile Class Reference

File operations class.

```
#include <BFile.h>
```

Inheritance diagram for BFile:



Public Member Functions

- BFile ()
- BFile (const BFile &file)

  Create opened specified file.
- ~BFile ()
- BError open (BString name, BString mode)

  Open file.
- BError open (FILE *file)

  Assign object to opened file handle.
- BError open (int fd, BString mode)

  Assign object to opened file descriptor.
- BError close ()

  Close file.
- int isOpen ()

  Returns 1 if the file is open.
- int isEnd ()

  Returns 1 if at the end of the file, 0 otherwise.
- FILE * getFd ()

  File descriptor.
- BUInt64 length ()

  File size in bytes.
- int setVBuf (char *buf, int mode, size_t size)

  Set stream buffering options.
- int read (void *buf, int nbytes)

  Read from file.
- int readString (BString &str)

  Read string. (ref fgets)
- char * fgets (char *buf, size_t size)
- int write (const void *buf, int nbytes)

  Write to file.
- int writeString (const BString &str)

  Write string to file.
- int seek (BUInt64 pos)

  Set seek position.
- BUInt64 position ()

  The files position.
- int printf (const char *fmt,...)

  Formated print into the file.
- BError truncate ()

  Truncate the file.
- BError flush ()

  Flush the file.
- BString fileName ()

  Return file name.
- BFile & operator= (const BFile &file)

7.36.1 Detailed Description

File operations class.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 BFile() [1/2]

```
BFile::BFile ( )
```

7.36.2.2 BFile() [2/2]

```
BFile::BFile (
            const BFile & file )
```

Create opened specified file.

7.36.2.3 ~BFile()

```
BFile::~BFile ( )
```

7.36.3 Member Function Documentation

7.36.3.1 open() [1/3]

```
BError BFile::open (
            BString name,
            BString mode )
```

Open file.

7.36.3.2 open() [2/3]

```
BError BFile::open (
            FILE * file )
```

Assign object to opened file handle.

7.36.3.3 open() [3/3]

```
BError BFile::open (
            int fd,
            BString mode )
```

Assign object to opened file descriptor.

7.36.3.4 close()

```
BError BFile::close ( )
```

Close file.

7.36.3.5 isOpen()

```
int BFile::isOpen ( )
```

Returns 1 if the file is open.

7.36.3.6 isEnd()

```
int BFile::isEnd ( )
```

Returns 1 if at the end of the file, 0 otherwise.

7.36.3.7 getFd()

```
FILE * BFile::getFd ( )
```

File descriptor.

7.36.3.8 length()

```
BUInt64 BFile::length ( )
```

File size in bytes.

7.36.3.9 setVBuf()

```
int BFile::setVBuf (
            char * buf,
            int mode,
            size_t size )
```

Set stream buffering options.

7.36.3.10 read()

```
int BFile::read (
            void * buf,
            int nbytes )
```

Read from file.

7.36.3.11 readString()

```
int BFile::readString (
            BString & str )
```

Read string. (ref fgets)

7.36.3.12 fgets()

```
char * BFile::fgets (
            char * buf,
            size_t size )
```

7.36.3.13 write()

```
int BFile::write (
            const void * buf,
            int nbytes )
```

Write to file.

7.36.3.14 writeString()

```
int BFile::writeString (
            const BString & str )
```

Write string to file.

7.36.3.15 seek()

```
int BFile::seek (
            BUInt64 pos )
```

Set seek position.

7.36.3.16 position()

```
BUInt64 BFile::position ( )
```

The files position.

7.36.3.17 printf()

```
int BFile::printf (
            const char * fmt,
             ...  )
```

Formated print into the file.

7.36.3.18 truncate()

```
BError BFile::truncate ( )
```

Truncate the file.

7.36.3.19 flush()

```
BError BFile::flush ( )
```

Flush the file.

7.36.3.20 fileName()

```
BString BFile::fileName ( )
```

Return file name.

7.36.3.21 operator=()

```
BFile & BFile::operator= (
            const BFile & file )
```

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

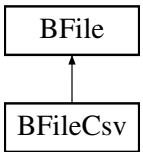
- BFile.h
- BFile.cpp

7.37 BFileCsv Class Reference

A class to read and write CSV formatted files.

```
#include <BFileCsv.h>
```

Inheritance diagram for BFileCsv:



Public Member Functions

- BFileCsv (char separator=';')
- BError readCsv (BStringList &csvList)
- BError writeCsv (BStringList &csvList)

7.37.1 Detailed Description

A class to read and write CSV formatted files.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 BFileCsv()

```
BFileCsv::BFileCsv (
            char separator = ';' )
```

7.37.3 Member Function Documentation

7.37.3.1 readCsv()

```
BError BFileCsv::readCsv (
            BStringList & csvList )
```

7.37.3.2 writeCsv()

```
BError BFileCsv::writeCsv (
            BStringList & csvList )
```

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

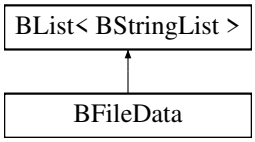
- BFileCsv.h
- BFileCsv.cpp

7.38 BFileData Class Reference

A class to implement a data storage file.

`#include <BFileData.h>`

Inheritance diagram for BFileData:



Public Member Functions

- BError open (BString filename)
- BError getNextId (int &id)
- BError find (int id, BStringList &csvList)
- BError write (int id, BStringList &csvList)
- BError del (int id)

Additional Inherited Members

7.38.1 Detailed Description

A class to implement a data storage file.

7.38.2 Member Function Documentation

7.38.2.1 open()

```
BError BFileData::open (
            BString filename )
```

7.38.2.2 getNextId()

```
BError BFileData::getNextId (
            int & id )
```

7.38.2.3 find()

```
BError BFileData::find (
            int id,
            BStringList & csvList )
```

7.38.2.4 write()

```
BError BFileData::write (
            int id,
            BStringList & csvList )
```

7.38.2.5 del()

```
BError BFileData::del (
            int id )
```

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

- BFileData.h
- BFileData.cpp

7.39 BFirmwareFileHeader Struct Reference

`#include <BFirmware.h>`

Public Attributes

- BUInt32 magic
- BUInt32 itemType
- BUInt32 fileLength
- BUInt32 checksum
- BUInt32 platform
- BUInt32 format
- BUInt32 numSegments
- BUInt32 startAddress
- BUInt8 ver0
- BUInt8 ver1
- BUInt8 ver2
- BUInt8 ver3
- BUInt32 special [7]

7.39.1 Member Data Documentation

7.39.1.1 magic

`BUInt32 BFirmwareFileHeader::magic`

7.39.1.2 itemType

`BUInt32 BFirmwareFileHeader::itemType`

7.39.1.3 fileLength

`BUInt32 BFirmwareFileHeader::fileLength`

7.39.1.4 checksum

`BUInt32 BFirmwareFileHeader::checksum`

7.39.1.5 platform

```
BUInt32 BFirmwareFileHeader::platform
```

7.39.1.6 format

```
BUInt32 BFirmwareFileHeader::format
```

7.39.1.7 numSegments

```
BUInt32 BFirmwareFileHeader::numSegments
```

7.39.1.8 startAddress

```
BUInt32 BFirmwareFileHeader::startAddress
```

7.39.1.9 ver0

```
BUInt8 BFirmwareFileHeader::ver0
```

7.39.1.10 ver1

```
BUInt8 BFirmwareFileHeader::ver1
```

7.39.1.11 ver2

```
BUInt8 BFirmwareFileHeader::ver2
```

7.39.1.12 ver3

```
BUInt8 BFirmwareFileHeader::ver3
```

7.39.1.13 special

```
BUInt32 BFirmwareFileHeader::special[7]
```

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

- BFirmware.h

7.40 BFirmwareInfo Struct Reference

```
#include <BFirmware.h>
```

Public Attributes

- BUInt32 magic
- BUInt32 length
- BUInt32 checksum
- BUInt8 type
- BUInt8 ver0
- BUInt8 ver1
- BUInt8 ver2

7.40.1 Member Data Documentation

7.40.1.1 magic

```
BUInt32 BFirmwareInfo::magic
```

7.40.1.2 length

```
BUInt32 BFirmwareInfo::length
```

7.40.1.3 checksum

```
BUInt32 BFirmwareInfo::checksum
```

7.40.1.4 type

```
BUInt8 BFirmwareInfo::type
```

7.40.1.5 ver0

```
BUInt8 BFirmwareInfo::ver0
```

7.40.1.6 ver1

```
BUInt8 BFirmwareInfo::ver1
```

7.40.1.7 ver2

```
BUInt8 BFirmwareInfo::ver2
```

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

- BFirmware.h

7.41 BFirmwareSegHeader Struct Reference

```
#include <BFirmware.h>
```

Public Attributes

- BUInt32 magic
- BUInt32 itemType
- BUInt32 fileLength
- BUInt32 checksum
- BUInt32 platform
- BUInt32 format
- BUInt32 dataLength
- BUInt32 address
- BUInt32 length
- BUInt32 special [7]

7.41.1 Member Data Documentation

7.41.1.1 magic

```
BUInt32 BFirmwareSegHeader::magic
```

7.41.1.2 itemType

```
BUInt32 BFirmwareSegHeader::itemType
```

7.41.1.3 fileLength

```
BUInt32 BFirmwareSegHeader::fileLength
```

7.41.1.4 checksum

```
BUInt32 BFirmwareSegHeader::checksum
```

7.41.1.5 platform

```
BUInt32 BFirmwareSegHeader::platform
```

7.41.1.6 format

```
BUInt32 BFirmwareSegHeader::format
```

7.41.1.7 dataLength

```
BUInt32 BFirmwareSegHeader::dataLength
```

7.41.1.8 address

```
BUInt32 BFirmwareSegHeader::address
```

7.41.1.9 length

```
BUInt32 BFirmwareSegHeader::length
```

7.41.1.10 special

```
BUInt32 BFirmwareSegHeader::special[7]
```

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

- BFirmware.h

7.42 BIter Class Reference

Iterator for BLists.

```
#include <BList.h>
```

Public Member Functions

- BIter (BNode *i=0)
- operator BNode * ()
- int operator== (const BIter &i)
- int valid ()

7.42.1 Detailed Description

Iterator for BLists.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 BIter()

```
BIter::BIter (
            BNode * i = 0 )  [inline]
```

7.42.3 Member Function Documentation

7.42.3.1 operator BNode ∗()

```
BIter::operator BNode * ( )  [inline]
```

7.42.3.2 operator==()

```
int BIter::operator== (
            const BIter & i )  [inline]
```

7.42.3.3 valid()

```
int BIter::valid ( )  [inline]
```

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

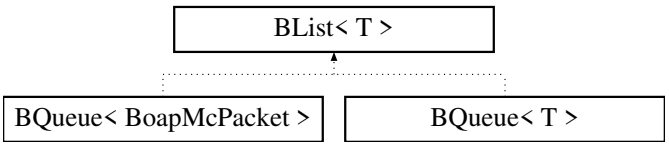
- BList.h

7.43 BList< T > Class Template Reference

Template based list class.

```
#include <BList.h>
```

Inheritance diagram for BList< T >:



Classes

- class Node

  A BList internal Node.

Public Types

- typedef int(∗ SortFunc) (T &a, T &b)

  Prototype for sorting function.

Public Member Functions

- BList ()
- BList (const BList< T > &l)
- virtual ~BList ()
- void start (BIter &i) const

  Iterator to start of list.
- BIter begin () const

  Iterator for start of list.
- BIter end () const

  Iterator for end of list.
- BIter end (BIter &i) const

  Iterator for end of list.
- void next (BIter &i) const

  Iterator for next item in list.
- void prev (BIter &i)

  Iterator for previous item in list.
- BIter goTo (int pos) const

  Iterator for pos item in list.
- int position (BIter i)

  Postition in list item with iterator i.
- unsigned int number () const

  Number of items in list.
- unsigned int size () const

  Number of items in list.
- int isStart (BIter &i) const

  True if iterator refers to first item.
- int isEnd (BIter &i) const

  True if iterator refers to last item.
- T & front ()

  Get first item in list.
- T & rear ()

  Get last item in list.
- T & get (BIter i)

  Get item specified by iterator in list.
- const T & get (BIter i) const

  Get item specified by iterator in list.
- void append (const T &item)

  Append item to list.
- virtual void insert (BIter &i, const T &item)

  Insert item before item.
- void insertAfter (BIter &i, const T &item)

  Insert item after item.
- virtual void clear ()

  Clear the list.
- virtual void del (BIter &i)

  Delete specified item.
- void deleteLast ()

  Delete last item.
- void deleteFirst ()

  Delete fisrt item.

- void push (const T &i)

  Push item onto list.
- T pop ()

  Pop item from list deleteing item.
- void queueAdd (const T &i)

  Add item to end of list.
- T queueGet ()

  Get item from front of list deleteing item.
- void append (const BList< T > &l)

  Append list to list.
- int has (const T &i) const

  Checks if the item is in the list.
- void swap (BIter i1, BIter i2)

  Swap two items in list.
- void sort ()

  Sort list based on get(i) values.
- void sort (SortFunc func)

  Sort list based on Sort func.
- BList< T > & operator= (const BList< T > &l)
- T & operator[ ] (int i)
- const T & operator[ ] (int i) const
- T & operator[ ] (BIter i)
- const T & operator[ ] (const BIter &i) const
- BList< T > operator+ (const BList< T > &l) const

Protected Member Functions

- virtual Node * nodeGet (BIter i)
- virtual const Node * nodeGet (BIter i) const
- virtual Node * nodeCreate (const T &item)

Protected Attributes

- Node * onodes
- unsigned int olength

7.43.1 Detailed Description

template<class T>
class BList< T >

Template based list class.

The BList class is a simple doubly linked list of objects. It is used to store an ordered list of any type/class of objects. The class provides a simple iteration system to allow easy navigation through the list. You can access objects stored in the list with the get() function or the [] operator. The list supports stack functions push() and pop() and queueing functions such as queueAdd() and queueGet(). There is a macro BListLoop(list, itterator) that provides a concise way to iterate over a lists objects.

7.43.2 Member Typedef Documentation

7.43.2.1 SortFunc

```
template<class T >
typedef int(* BList< T >::SortFunc) (T &a, T &b)
```

Prototype for sorting function.

7.43.3 Constructor & Destructor Documentation

7.43.3.1 BList() [1/2]

```
template<class T >
BList< T >::BList
```

7.43.3.2 BList() [2/2]

```
template<class T >
BList< T >::BList (
            const BList< T > & l )
```

7.43.3.3 ~BList()

```
template<class T >
BList< T >::~BList  [virtual]
```

7.43.4 Member Function Documentation

7.43.4.1 start()

```
template<class T >
void BList< T >::start (
            BIter & i ) const
```

Iterator to start of list.

7.43.4.2 begin()

```
template<class T >
BIter BList< T >::begin
```

Iterator for start of list.

7.43.4.3 end() [1/2]

```
template<class T >
BIter BList< T >::end
```

Iterator for end of list.

7.43.4.4 end() [2/2]

```
template<class T >
BIter BList< T >::end (
            BIter & i ) const
```

Iterator for end of list.

7.43.4.5 next()

```
template<class T >
void BList< T >::next (
            BIter & i ) const
```

Iterator for next item in list.

7.43.4.6 prev()

```
template<class T >
void BList< T >::prev (
            BIter & i )
```

Iterator for previous item in list.

7.43.4.7 goTo()

```
template<class T >
BIter BList< T >::goTo (
            int pos ) const
```

Iterator for pos item in list.

7.43.4.8 position()

```
template<class T >
int BList< T >::position (
            BIter i )
```

Postition in list item with iterator i.

7.43.4.9 number()

```
template<class T >
unsigned int BList< T >::number
```

Number of items in list.

7.43.4.10 size()

```
template<class T >
unsigned int BList< T >::size
```

Number of items in list.

7.43.4.11 isStart()

```
template<class T >
int BList< T >::isStart (
            BIter & i ) const
```

True if iterator refers to first item.

7.43.4.12 isEnd()

```
template<class T >
int BList< T >::isEnd (
            BIter & i ) const
```

True if iterator refers to last item.

7.43.4.13 front()

```
template<class T >
T & BList< T >::front
```

Get first item in list.

7.43.4.14 rear()

```
template<class T >
T & BList< T >::rear
```

Get last item in list.

7.43.4.15 get() [1/2]

```
template<class T >
T & BList< T >::get (
            BIter i )
```

Get item specified by iterator in list.

7.43.4.16 get() [2/2]

```
template<class T >
const T & BList< T >::get (
            BIter i ) const
```

Get item specified by iterator in list.

7.43.4.17 append() [1/2]

```
template<class T >
void BList< T >::append (
            const T & item )
```

Append item to list.

7.43.4.18 insert()

```
template<class T >
void BList< T >::insert (
            BIter & i,
            const T & item )  [virtual]
```

Insert item before item.

7.43.4.19 insertAfter()

```
template<class T >
void BList< T >::insertAfter (
            BIter & i,
            const T & item )
```

Insert item after item.

7.43.4.20 clear()

```
template<class T >
void BList< T >::clear  [virtual]
```

Clear the list.

Reimplemented in BQueue< T >, BQueue< BoapMcPacket >, BEntryFile, BEntryList, BDir, and BDict< Type >.

7.43.4.21 del()

```
template<class T >
void BList< T >::del (
            BIter & i )  [virtual]
```

Delete specified item.

Reimplemented in BEntryList, and BDict< Type >.

7.43.4.22 deleteLast()

```
template<class T >
void BList< T >::deleteLast
```

Delete last item.

7.43.4.23 deleteFirst()

```
template<class T >
void BList< T >::deleteFirst
```

Delete fisrt item.

7.43.4.24 push()

```
template<class T >
void BList< T >::push (
            const T & i )
```

Push item onto list.

7.43.4.25 pop()

```
template<class T >
T BList< T >::pop
```

Pop item from list deleteing item.

7.43.4.26 queueAdd()

```
template<class T >
void BList< T >::queueAdd (
            const T & i )
```

Add item to end of list.

7.43.4.27 queueGet()

```
template<class T >
T BList< T >::queueGet
```

Get item from front of list deleteing item.

7.43.4.28 append() [2/2]

```
template<class T >
void BList< T >::append (
            const BList< T > & l )
```

Append list to list.

7.43.4.29 has()

```
template<class T >
int BList< T >::has (
            const T & i ) const
```

Checks if the item is in the list.

7.43.4.30 swap()

```
template<class T >
void BList< T >::swap (
            BIter i1,
            BIter i2 )
```

Swap two items in list.

7.43.4.31 sort() [1/2]

```
template<class T >
void BList< T >::sort
```

Sort list based on get(i) values.

7.43.4.32 sort() [2/2]

```
template<class T >
void BList< T >::sort (
            SortFunc func )
```

Sort list based on Sort func.

7.43.4.33 operator=()

```
template<class T >
BList< T > & BList< T >::operator= (
            const BList< T > & l )
```

7.43.4.34 operator[]() [1/4]

```
template<class T >
T & BList< T >::operator[] (
            int i )
```

7.43.4.35 operator[]() [2/4]

```
template<class T >
const T & BList< T >::operator[] (
            int i ) const
```

7.43.4.36 operator[]() [3/4]

```
template<class T >
T & BList< T >::operator[] (
            BIter i )
```

7.43.4.37 operator[]() [4/4]

```
template<class T >
const T & BList< T >::operator[] (
            const BIter & i ) const
```

7.43.4.38 operator+()

```
template<class T >
BList< T > BList< T >::operator+ (
            const BList< T > & l ) const
```

7.43.4.39 nodeGet() [1/2]

```
template<class T >
BList< T >::Node * BList< T >::nodeGet (
            BIter i )  [protected], [virtual]
```

7.43.4.40 nodeGet() [2/2]

```
template<class T >
const BList< T >::Node * BList< T >::nodeGet (
            BIter i ) const  [protected], [virtual]
```

7.43.4.41 nodeCreate()

```
template<class T >
BList< T >::Node * BList< T >::nodeCreate (
            const T & item )  [protected], [virtual]
```

7.43.5 Member Data Documentation

7.43.5.1 onodes

```
template<class T >
Node* BList< T >::onodes  [protected]
```

7.43.5.2 olength

```
template<class T >
unsigned int BList< T >::olength  [protected]
```

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

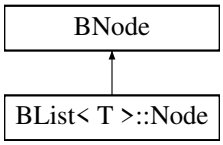
- BList.h
- BList_func.h

7.44 BList< T >::Node Class Reference

A BList internal Node.

`#include <BList.h>`

Inheritance diagram for BList< T >::Node:



Public Member Functions

- Node (const T &i)

Public Attributes

- T item

7.44.1 Detailed Description

template<class T>
class BList< T >::Node

A BList internal Node.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 Node()

```
template<class T >
BList< T >::Node::Node (
            const T & i )  [inline]
```

7.44.3 Member Data Documentation

7.44.3.1 item

```
template<class T >
T BList< T >::Node::item
```

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

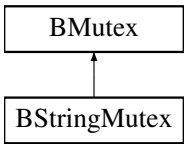
- BList.h

7.45 BMutex Class Reference

Mutex class. Note these are recursive Mutexes and so you need to make sure the number of unlocks equals the number of locks.

`#include <BMutex.h>`

Inheritance diagram for BMutex:



Public Types

- enum Type { Normal , Recursive }

Public Member Functions

- BMutex (Type type=Normal)
- BMutex (const BMutex &mutex)
- ~BMutex ()
- int lock ()

  Set lock, wait as necessary.
- int timedLock (int timeoutUs)

  Set lock, wait as necessary but timeout after given time.
- int unlock ()

  Unlock the lock.
- int tryLock ()

  Test the lock.
- BMutex & operator= (const BMutex &mutex)

7.45.1 Detailed Description

Mutex class. Note these are recursive Mutexes and so you need to make sure the number of unlocks equals the number of locks.

7.45.2 Member Enumeration Documentation

7.45.2.1 Type

```
enum BMutex::Type
```

Enumerator

Normal	
Recursive	

7.45.3 Constructor & Destructor Documentation

7.45.3.1 BMutex() [1/2]

```
BMutex::BMutex (
            Type type = Normal )
```

7.45.3.2 BMutex() [2/2]

```
BMutex::BMutex (
            const BMutex & mutex )
```

7.45.3.3 ~BMutex()

```
BMutex::~BMutex ( )
```

7.45.4 Member Function Documentation

7.45.4.1 lock()

```
int BMutex::lock ( )
```

Set lock, wait as necessary.

7.45.4.2 timedLock()

```
int BMutex::timedLock (
            int timeoutUs )
```

Set lock, wait as necessary but timeout after given time.

7.45.4.3 unlock()

```
int BMutex::unlock ( )
```

Unlock the lock.

7.45.4.4 tryLock()

```
int BMutex::tryLock ( )
```

Test the lock.

7.45.4.5 operator=()

```
BMutex & BMutex::operator= (
            const BMutex & mutex )
```

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

- BMutex.h
- BMutex.cpp

7.46 BMutexLock Class Reference

Mutex class that removes the lock on deletion and so is useful to lock data in a function call.

```
#include <BMutex.h>
```

Public Member Functions

- BMutexLock (BMutex &lock, int doLock=0)
- ~BMutexLock ()
- int lock ()
- int unlock ()

7.46.1 Detailed Description

Mutex class that removes the lock on deletion and so is useful to lock data in a function call.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 BMutexLock()

```
BMutexLock::BMutexLock (
            BMutex & lock,
            int doLock = 0 )  [inline]
```

7.46.2.2 ~BMutexLock()

```
BMutexLock::~BMutexLock ( )  [inline]
```

7.46.3 Member Function Documentation

7.46.3.1 lock()

```
int BMutexLock::lock ( )  [inline]
```

7.46.3.2 unlock()

```
int BMutexLock::unlock ( )  [inline]
```

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

- BMutex.h

7.47 BMysql Class Reference

A class to provide access to a MySQL database.

`#include <BMysql.h>`

Public Member Functions

- BMysql ()
- ~BMysql ()
- BError open (BString hostName, BString dataBase, BString userName, BString password)
- void close ()
- BError get (BString table, BString where, BDictString &fields)
- BError insert (BString table, BDictString fields, BUInt32 *id=0)
- BError update (BString table, BUInt32 id, BDictString fields)
- BError del (BString table, BUInt32 id)

  Delete record from table.
- BError flush ()

  Flush all data to disk.
- BString escapeString (BString str)

  Escapes special characters in the string.
- BError query (BString cmd, BList< BDictString > &result)
- MYSQL & db ()
- void setDebug (int debug)

7.47.1 Detailed Description

A class to provide access to a MySQL database.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 BMysql()

```
BMysql::BMysql ( )
```

7.47.2.2 ~BMysql()

```
BMysql::~BMysql ( )
```

7.47.3 Member Function Documentation

7.47.3.1 open()

```
BError BMysql::open (
            BString hostName,
            BString dataBase,
            BString userName,
            BString password )
```

7.47.3.2 close()

```
void BMysql::close ( )
```

7.47.3.3 get()

```
BError BMysql::get (
            BString table,
            BString where,
            BDictString & fields )
```

7.47.3.4 insert()

```
BError BMysql::insert (
            BString table,
            BDictString fields,
            BUInt32 * id = 0 )
```

7.47.3.5 update()

```
BError BMysql::update (
            BString table,
            BUInt32 id,
            BDictString fields )
```

7.47.3.6 del()

```
BError BMysql::del (
            BString table,
            BUInt32 id )
```

Delete record from table.

7.47.3.7 flush()

```
BError BMysql::flush ( )
```

Flush all data to disk.

7.47.3.8 escapeString()

```
BString BMysql::escapeString (
            BString str )
```

Escapes special characters in the string.

7.47.3.9 query()

```
BError BMysql::query (
            BString cmd,
            BList< BDictString > & result )
```

7.47.3.10 db()

```
MYSQL & BMysql::db ( )
```

7.47.3.11 setDebug()

```
void BMysql::setDebug (
            int debug )
```

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

- BMysql.h
- BMysql.cpp

7.48 BNameValue< T > Class Template Reference

A simple, templated, name/value pair.

```
#include <BNameValue.h>
```

Public Member Functions

- BNameValue ()
- BNameValue (BString name, const T &value)
- BString getName ()
- T & getValue ()

7.48.1 Detailed Description

template<class T>
class BNameValue< T >

A simple, templated, name/value pair.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 BNameValue() [1/2]

```
template<class T >
BNameValue< T >::BNameValue ( )  [inline]
```

7.48.2.2 BNameValue() [2/2]

```
template<class T >
BNameValue< T >::BNameValue (
            BString name,
            const T & value )  [inline]
```

7.48.3 Member Function Documentation

7.48.3.1 getName()

```
template<class T >
BString BNameValue< T >::getName ( )  [inline]
```

7.48.3.2 getValue()

```
template<class T >
T& BNameValue< T >::getValue ( )  [inline]
```

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

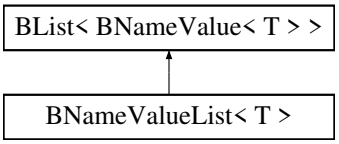
- BNameValue.h

7.49 BNameValueList< T > Class Template Reference

A simple, templated, name/value pair list.

`#include <BNameValue.h>`

Inheritance diagram for BNameValueList< T >:



Public Member Functions

- T * find (BString name)
- BIter findPos (BString name)

Additional Inherited Members

7.49.1 Detailed Description

template<class T>
class BNameValueList< T >

A simple, templated, name/value pair list.

7.49.2 Member Function Documentation

7.49.2.1 find()

```
template<class T >
T* BNameValueList< T >::find (
            BString name )  [inline]
```

7.49.2.2 findPos()

```
template<class T >
BIter BNameValueList< T >::findPos (
            BString name )  [inline]
```

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

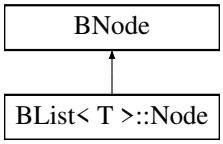
- BNameValue.h

7.50 BNode Class Reference

A BList entry's node.

```
#include <BList.h>
```

Inheritance diagram for BNode:



Public Member Functions

- BNode ()

Public Attributes

- BNode * next
- BNode * prev

7.50.1 Detailed Description

A BList entry's node.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 BNode()

```
BNode::BNode ( )  [inline]
```

7.50.3 Member Data Documentation

7.50.3.1 next

```
BNode* BNode::next
```

7.50.3.2 prev

```
BNode* BNode::prev
```

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

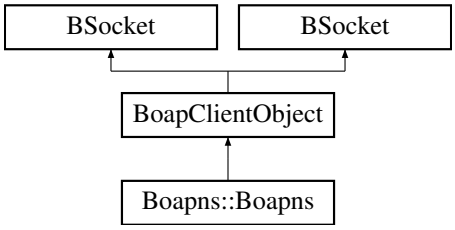
- BList.h

7.51 BoapClientObject Class Reference

Base for all Boap client objects.

`#include <Boap.h>`

Inheritance diagram for BoapClientObject:



Public Member Functions

- BoapClientObject (BString name="")
- virtual ~BoapClientObject ()
- BUInt32 apiVersion ()

  Returns the API version.
- BError connectService (BString name)

  Connects to the named service.
- BError disconnectService ()

  Disconnects from the named service.
- BString getServiceName ()

  Get the name of the service.
- BError ping (BUInt32 &apiVersion)

  Pings the connection and finds the remotes version number.
- BError setConnectionPriority (BoapPriority priority)

  Sets the connection priority.
- void setMaxLength (BUInt32 maxLength)

  Sets the maximum packet length.
- void setTimeout (int timeout)

  Sets the timeout in micro seconds. -1 is wait indefinitely.
- BoapClientObject (BString name)
- BError connectService (BString name)

Protected Member Functions

- BError pingLocked (BUInt32 &apiVersion)
- BError checkApiVersion ()
- BError performCall (BoapPacket &tx, BoapPacket &rx)

  Performs a RPC call to the named service.
- BError performSend (BoapPacket &tx)

  Performs a send to the named service.
- BError performRecv (BoapPacket &rx)

  Performs a receive.
- virtual BError handleReconnect (BError err)

  Handle a reconnect performing autorisaztion if required.
- BError performSend (BoapPacket &tx)
- BError performRecv (BoapPacket &rx)
- BError performCall (BoapPacket &tx, BoapPacket &rx)

Protected Attributes

- BString oname
- BUInt32 oapiVersion
- BoapPriority opriority
- BoapService oservice
- int oconnected
- BUInt32 omaxLength
- BoapPacket otx
- BoapPacket orx
- BMutex olock
- int otimeout
- int oreconnect

  Handle an automatic reconnect on timeout.

Additional Inherited Members

7.51.1 Detailed Description

Base for all Boap client objects.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 BoapClientObject() [1/2]

```
BoapClientObject::BoapClientObject (
            BString name = "" )
```

7.51.2.2 ~BoapClientObject()

```
BoapClientObject::~BoapClientObject ( )  [virtual]
```

7.51.2.3 BoapClientObject() [2/2]

```
BoapClientObject::BoapClientObject (
            BString name )
```

7.51.3 Member Function Documentation

7.51.3.1 apiVersion()

```
BUInt32 BoapClientObject::apiVersion ( )
```

Returns the API version.

7.51.3.2 connectService() [1/2]

```
BError BoapClientObject::connectService (
            BString name )
```

Connects to the named service.

7.51.3.3 disconnectService()

```
BError BoapClientObject::disconnectService ( )
```

Disconnects from the named service.

7.51.3.4 getServiceName()

```
BString BoapClientObject::getServiceName ( )
```

Get the name of the service.

7.51.3.5 ping()

```
BError BoapClientObject::ping (
            BUInt32 & apiVersion )
```

Pings the connection and finds the remotes version number.

7.51.3.6 setConnectionPriority()

```
BError BoapClientObject::setConnectionPriority (
            BoapPriority priority )
```

Sets the connection priority.

7.51.3.7 setMaxLength()

```
void BoapClientObject::setMaxLength (
            BUInt32 maxLength )
```

Sets the maximum packet length.

7.51.3.8 setTimeout()

```
void BoapClientObject::setTimeout (
            int timeout )
```

Sets the timeout in micro seconds. -1 is wait indefinitely.

7.51.3.9 pingLocked()

```
BError BoapClientObject::pingLocked (
            BUInt32 & apiVersion )  [protected]
```

7.51.3.10 checkApiVersion()

```
BError BoapClientObject::checkApiVersion ( )  [protected]
```

7.51.3.11 performCall() [1/2]

```
BError BoapClientObject::performCall (
            BoapPacket & tx,
            BoapPacket & rx )  [protected]
```

Performs a RPC call to the named service.

7.51.3.12 performSend() [1/2]

```
BError BoapClientObject::performSend (
            BoapPacket & tx )  [protected]
```

Performs a send to the named service.

7.51.3.13 performRecv() [1/2]

```
BError BoapClientObject::performRecv (
            BoapPacket & rx )  [protected]
```

Performs a receive.

7.51.3.14 handleReconnect()

```
BError BoapClientObject::handleReconnect (
            BError err )  [protected], [virtual]
```

Handle a reconnect performing autorisaztion if required.

7.51.3.15 connectService() [2/2]

```
BError BoapClientObject::connectService (
            BString name )
```

7.51.3.16 performSend() [2/2]

```
BError BoapClientObject::performSend (
            BoapPacket & tx )  [protected]
```

7.51.3.17 performRecv() [2/2]

```
BError BoapClientObject::performRecv (
            BoapPacket & rx )  [protected]
```

7.51.3.18 performCall() [2/2]

```
BError BoapClientObject::performCall (
            BoapPacket & tx,
            BoapPacket & rx )  [protected]
```

7.51.4 Member Data Documentation

7.51.4.1 oname

BString BoapClientObject::oname [protected]

7.51.4.2 oapiVersion

BUInt32 BoapClientObject::oapiVersion [protected]

7.51.4.3 opriority

BoapPriority BoapClientObject::opriority [protected]

7.51.4.4 oservice

BoapService BoapClientObject::oservice [protected]

7.51.4.5 oconnected

int BoapClientObject::oconnected [protected]

7.51.4.6 omaxLength

BUInt32 BoapClientObject::omaxLength [protected]

7.51.4.7 otx

BoapPacket BoapClientObject::otx [protected]

7.51.4.8 orx

BoapPacket BoapClientObject::orx [protected]

7.51.4.9 olock

```
BMutex BoapClientObject::olock  [protected]
```

7.51.4.10 otimeout

```
int BoapClientObject::otimeout  [protected]
```

7.51.4.11 oreconnect

```
int BoapClientObject::oreconnect  [protected]
```

Handle an automatic reconnect on timeout.

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

- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.52 BoapFuncEntry Class Reference

Boap service function.

```
#include <Boap.h>
```

Public Member Functions

- BoapFuncEntry (int cmd, BoapFunc func)
- BoapFuncEntry (int cmd, BoapFunc func)

Public Attributes

- BUInt32 ocmd
- BoapFunc ofunc
- UInt32 ocmd

7.52.1 Detailed Description

Boap service function.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 BoapFuncEntry() [1/2]

```
BoapFuncEntry::BoapFuncEntry (
            int cmd,
            BoapFunc func )
```

7.52.2.2 BoapFuncEntry() [2/2]

```
BoapFuncEntry::BoapFuncEntry (
            int cmd,
            BoapFunc func )
```

7.52.3 Member Data Documentation

7.52.3.1 ocmd [1/2]

```
BUInt32 BoapFuncEntry::ocmd
```

7.52.3.2 ofunc

```
BoapFunc BoapFuncEntry::ofunc
```

7.52.3.3 ocmd [2/2]

```
UInt32 BoapFuncEntry::ocmd
```

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

- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.53 BoapMc1Comms Class Reference

`#include <BoapMc1.h>`

Public Member Functions

- BoapMc1Comms (Bool threaded=0, BUInt reqSize=512)
- virtual ~BoapMc1Comms ()
- void setCommsMode (Bool halfDuplex)

  Sets half duplex mode.
- void setComms (BComms &comms)

  Sets the communications interface to use.
- void setComms (BComms *comms)

  Sets the communications interface to use.
- void setAddress (BUInt16 addressTo, BUInt16 addressFrom)

  Sets the to and from addresses.
- BUInt32 getApiVersion ()

  Returns the API version.
- BUInt32 setTimeout (BUInt32 timeoutUs)

  Sets the call timeout returning the current value.
- virtual BError validate ()

  Validate the request.
- BoapMc1Packet * packetRx ()

  Returns a reference to the current RX packet.
- virtual BError processRx ()

  Process any RX packets queuing them as needed.

Protected Member Functions

- virtual BError processRequests ()

  Check and process any requests.
- virtual BError processRequest ()

  Check and process any request.
- BError packetTx (BDataChunk *chunks, BUInt nChunks, BUInt16 waitCmdReply)
- BError packetRxData (void *data, BUInt nBytes)
- BError packetRxEnd ()

Protected Attributes

- Bool othreaded

  Threaded operation.
- BUInt32 oreqSize

  The maximum request size.
- BMutex olockCall

  Lock for RPC calls. Only one at a time.
- BMutex olockTx

  Lock for TX.
- BComms * ocomms
- BUInt32 oapiVersion

- Bool ohalfDuplex

  Half duplex mode.
- BUInt32 otimeout

  The timeout in us for calls.
- BUInt16 oaddressTo
- BUInt16 oaddressFrom
- BoapMc1Packet opacketRxBase
- BoapMc1Packet * opacketRx

  The RX packet.
- BoapMc1Packet opacketTxBase
- BoapMc1Packet * opacketTx

  The TX packet.
- BUInt opacketRpcCmd

  Waiting for RPC reply to cmd.
- BSemaphore opacketRpcSema

  Wait RPC reply semaphore.
- BSemaphore opacketRpcDoneSema

  Wait RPC complete semaphore.
- BoapMc1Error oerror

  The call return error;.

7.53.1 Constructor & Destructor Documentation

7.53.1.1 BoapMc1Comms()

```
BoapMc1Comms::BoapMc1Comms (
            Bool threaded = 0,
            BUInt reqSize = 512 )
```

7.53.1.2 ~BoapMc1Comms()

```
BoapMc1Comms::~BoapMc1Comms ( )  [virtual]
```

7.53.2 Member Function Documentation

7.53.2.1 setCommsMode()

```
void BoapMc1Comms::setCommsMode (
            Bool halfDuplex )
```

Sets half duplex mode.

7.53.2.2 setComms() [1/2]

```
void BoapMc1Comms::setComms (
            BComms & comms )
```

Sets the communications interface to use.

7.53.2.3 setComms() [2/2]

```
void BoapMc1Comms::setComms (
            BComms * comms )
```

Sets the communications interface to use.

7.53.2.4 setAddress()

```
void BoapMc1Comms::setAddress (
            BUInt16 addressTo,
            BUInt16 addressFrom )
```

Sets the to and from addresses.

7.53.2.5 getApiVersion()

```
BUInt32 BoapMc1Comms::getApiVersion ( )
```

Returns the API version.

7.53.2.6 setTimeout()

```
BUInt32 BoapMc1Comms::setTimeout (
            BUInt32 timeoutUs )
```

Sets the call timeout returning the current value.

7.53.2.7 validate()

```
BError BoapMc1Comms::validate ( )  [virtual]
```

Validate the request.

7.53.2.8 packetRx()

```
BoapMc1Packet * BoapMc1Comms::packetRx ( )
```

Returns a reference to the current RX packet.

7.53.2.9 processRx()

```
BError BoapMc1Comms::processRx ( )  [virtual]
```

Process any RX packets queuing them as needed.

7.53.2.10 processRequests()

```
BError BoapMc1Comms::processRequests ( )  [protected], [virtual]
```

Check and process any requests.

7.53.2.11 processRequest()

```
BError BoapMc1Comms::processRequest ( )  [protected], [virtual]
```

Check and process any request.

7.53.2.12 packetTx()

```
BError BoapMc1Comms::packetTx (
            BDataChunk * chunks,
            BUInt nChunks,
            BUInt16 waitCmdReply )  [protected]
```

7.53.2.13 packetRxData()

```
BError BoapMc1Comms::packetRxData (
            void * data,
            BUInt nBytes )  [protected]
```

7.53.2.14 packetRxEnd()

```
BError BoapMc1Comms::packetRxEnd ( )  [protected]
```

7.53.3 Member Data Documentation

7.53.3.1 othreaded

```
Bool BoapMc1Comms::othreaded  [protected]
```

Threaded operation.

7.53.3.2 oreqSize

```
BUInt32 BoapMc1Comms::oreqSize  [protected]
```

The maximum request size.

7.53.3.3 olockCall

```
BMutex BoapMc1Comms::olockCall  [protected]
```

Lock for RPC calls. Only one at a time.

7.53.3.4 olockTx

```
BMutex BoapMc1Comms::olockTx  [protected]
```

Lock for TX.

7.53.3.5 ocomms

```
BComms* BoapMc1Comms::ocomms  [protected]
```

7.53.3.6 oapiVersion

```
BUInt32 BoapMc1Comms::oapiVersion  [protected]
```

7.53.3.7 ohalfDuplex

```
Bool BoapMc1Comms::ohalfDuplex  [protected]
```

Half duplex mode.

7.53.3.8 otimeout

```
BUInt32 BoapMc1Comms::otimeout  [protected]
```

The timeout in us for calls.

7.53.3.9 oaddressTo

```
BUInt16 BoapMc1Comms::oaddressTo  [protected]
```

7.53.3.10 oaddressFrom

```
BUInt16 BoapMc1Comms::oaddressFrom  [protected]
```

7.53.3.11 opacketRxBase

```
BoapMc1Packet BoapMc1Comms::opacketRxBase  [protected]
```

7.53.3.12 opacketRx

```
BoapMc1Packet* BoapMc1Comms::opacketRx  [protected]
```

The RX packet.

7.53.3.13 opacketTxBase

```
BoapMc1Packet BoapMc1Comms::opacketTxBase  [protected]
```

7.53.3.14 opacketTx

```
BoapMc1Packet* BoapMc1Comms::opacketTx  [protected]
```

The TX packet.

7.53.3.15 opacketRpcCmd

```
BUInt BoapMc1Comms::opacketRpcCmd  [protected]
```

Waiting for RPC reply to cmd.

7.53.3.16 opacketRpcSema

```
BSemaphore BoapMc1Comms::opacketRpcSema  [protected]
```

Wait RPC reply semaphore.

7.53.3.17 opacketRpcDoneSema

```
BSemaphore BoapMc1Comms::opacketRpcDoneSema  [protected]
```

Wait RPC complete semaphore.

7.53.3.18 oerror

```
BoapMc1Error BoapMc1Comms::oerror  [protected]
```

The call return error;.

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

- BoapMc1.h
- BoapMc1.cpp

7.54 BoapMc1Error Struct Reference

`#include <BoapMc1.h>`

Public Attributes

- BInt16 number
  The error number.
- char string [32]
  The error string.

7.54.1 Member Data Documentation

7.54.1.1 number

`BInt16 BoapMc1Error::number`

The error number.

7.54.1.2 string

`char BoapMc1Error::string[32]`

The error string.

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

- BoapMc1.h

7.55 BoapMc1Packet Class Reference

`#include <BoapMc1.h>`

Public Attributes

- BoapMc1PacketHead head
- char data [8]

7.55.1 Member Data Documentation

7.55.1.1 head

```
BoapMc1PacketHead BoapMc1Packet::head
```

7.55.1.2 data

```
char BoapMc1Packet::data[8]
```

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

- BoapMc1.h

7.56 BoapMc1PacketHead Struct Reference

```
#include <BoapMc1.h>
```

Public Attributes

- BUInt16 magic

  Packet magic pattern.
- BUInt16 length

  Total packet length including the header.
- BUInt16 addressTo

  Address to send to.
- BUInt16 addressFrom

  Address packet is from.
- BUInt16 cmd

  The RPC command or reply number.
- BInt16 error

  Error number.
- BUInt32 checksum

  Packet checksum, when used.

7.56.1 Member Data Documentation

7.56.1.1 magic

`BUInt16 BoapMc1PacketHead::magic`

Packet magic pattern.

7.56.1.2 length

`BUInt16 BoapMc1PacketHead::length`

Total packet length including the header.

7.56.1.3 addressTo

`BUInt16 BoapMc1PacketHead::addressTo`

Address to send to.

7.56.1.4 addressFrom

`BUInt16 BoapMc1PacketHead::addressFrom`

Address packet is from.

7.56.1.5 cmd

`BUInt16 BoapMc1PacketHead::cmd`

The RPC command or reply number.

7.56.1.6 error

`BInt16 BoapMc1PacketHead::error`

Error number.

7.56.1.7 checksum

```
BUInt32 BoapMc1PacketHead::checksum
```

Packet checksum, when used.

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

- BoapMc1.h

7.57 BoapMcClientObject Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcClientObject (BComms &comms)
- virtual ~BoapMcClientObject ()
- void setAddress (BUInt8 addressTo, BUInt8 addressFrom)
- BUInt32 getApiVersion ()

  Returns the API version.

Protected Member Functions

- BError performCall ()

  Performs a RPC call to the named service.
- BError performSend ()

  Performs a send to the named service.
- BError performRecv ()

  Performs a receive.

Protected Attributes

- BUInt32 oapiVersion
- BComms & ocomms
- BUInt8 oaddressTo
- BUInt8 oaddressFrom
- BoapMcPacket opacket

7.57.1 Constructor & Destructor Documentation

7.57.1.1 BoapMcClientObject()

```
BoapMcClientObject::BoapMcClientObject (
            BComms & comms )
```

7.57.1.2 ~BoapMcClientObject()

```
BoapMcClientObject::~BoapMcClientObject ( )  [virtual]
```

7.57.2 Member Function Documentation

7.57.2.1 setAddress()

```
void BoapMcClientObject::setAddress (
            BUInt8 addressTo,
            BUInt8 addressFrom )
```

7.57.2.2 getApiVersion()

```
BUInt32 BoapMcClientObject::getApiVersion ( )
```

Returns the API version.

7.57.2.3 performCall()

```
BError BoapMcClientObject::performCall ( )  [protected]
```

Performs a RPC call to the named service.

7.57.2.4 performSend()

```
BError BoapMcClientObject::performSend ( )  [protected]
```

Performs a send to the named service.

7.57.2.5 performRecv()

```
BError BoapMcClientObject::performRecv ( )  [protected]
```

Performs a receive.

7.57.3 Member Data Documentation

7.57.3.1 oapiVersion

```
BUInt32 BoapMcClientObject::oapiVersion  [protected]
```

7.57.3.2 ocomms

```
BComms& BoapMcClientObject::ocomms  [protected]
```

7.57.3.3 oaddressTo

```
BUInt8 BoapMcClientObject::oaddressTo  [protected]
```

7.57.3.4 oaddressFrom

```
BUInt8 BoapMcClientObject::oaddressFrom  [protected]
```

7.57.3.5 opacket

```
BoapMcPacket BoapMcClientObject::opacket  [protected]
```

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

- BoapMc.h
- BoapMc.cpp

7.58 BoapMcComms Class Reference

`#include <BoapMc.h>`

Public Member Functions

- BoapMcComms (Bool threaded=0, BUInt rxQueueSize=4)
- virtual ~BoapMcComms ()
- void setCommsMode (Bool slave, BUInt txQueueSize)

  Sets slave mode.
- void setComms (BComms &comms)

  Sets the communications interface to use.
- void setComms (BComms *comms)

  Sets the communications interface to use.
- void setAddress (BUInt8 addressTo, BUInt8 addressFrom)

  Sets the to and from addresses.
- BUInt32 getApiVersion ()

  Returns the API version.
- BUInt32 setTimeout (BUInt32 timeoutUs)

  Sets the call timeout returning the current value.
- virtual BError processRx (BTimeout timeoutUs=BTimeoutForever)

  Process any RX packets queuing them as needed.
- virtual BError processRequests (BTimeout timeoutUs=BTimeoutForever)

  Check and process all requests.
- virtual BError processRequest (BTimeout timeoutUs=BTimeoutForever)

  Check and process any request.
- virtual BError processPacket (BoapMcPacket &rx, BoapMcPacket &tx)

  Process a recieved packet.

Protected Member Functions

- BError performCall ()

  Performs a RPC call to the remote side.
- BError performSend ()

  Performs a RPC send to the remote side.
- BError packetSend (BoapMcPacket &packet)

  Receives a packet.
- BError packetRecv (BoapMcPacket &packet)

  Receives a packet.

Protected Attributes

- Bool othreaded
- BMutex olockCall

  Lock for RPC calls. Only one at a time.
- BMutex olockTx

  Lock for TX.
- BComms * ocomms
- BUInt32 oapiVersion
- Bool oslave

  Set slave mode.
- BUInt32 otimeout

  The timeout in us for calls.
- BUInt8 oaddressTo
- BUInt8 oaddressFrom
- BoapMcPacket opacket

  Packet RX buffer.
- BoapMcPacket opacketTx

  Packet TX buffer for calls.
- BoapMcPacket opacketRx

  Packet RX buffer for calls.
- BSemaphore opacketRxSema

  Wait RX semaphore.
- BoapMcPacket opacketReqTx

  Packet TX buffer for requests.
- BoapMcPacket opacketReqRx

  Packet RX buffer for requests.
- BQueue< BoapMcPacket > opacketReqQueue

  Packet RX buffer queue for requests.
- BFifo< BoapMcPacket > opacketTxQueue

  Packet TX Queue.
- BSemaphoreCount opacketTxQueueWriteNum

  Packet TX Queue number.
- BSemaphore opacketTxSema

  Wait for TX semaphore.

7.58.1 Constructor & Destructor Documentation

7.58.1.1 BoapMcComms()

```
BoapMcComms::BoapMcComms (
            Bool threaded = 0,
            BUInt rxQueueSize = 4 )
```

7.58.1.2 ~BoapMcComms()

```
BoapMcComms::~BoapMcComms ( )  [virtual]
```

7.58.2 Member Function Documentation

7.58.2.1 setCommsMode()

```
void BoapMcComms::setCommsMode (
            Bool slave,
            BUInt txQueueSize )
```

Sets slave mode.

7.58.2.2 setComms() [1/2]

```
void BoapMcComms::setComms (
            BComms & comms )
```

Sets the communications interface to use.

7.58.2.3 setComms() [2/2]

```
void BoapMcComms::setComms (
            BComms * comms )
```

Sets the communications interface to use.

7.58.2.4 setAddress()

```
void BoapMcComms::setAddress (
            BUInt8 addressTo,
            BUInt8 addressFrom )
```

Sets the to and from addresses.

7.58.2.5 getApiVersion()

```
BUInt32 BoapMcComms::getApiVersion ( )
```

Returns the API version.

7.58.2.6 setTimeout()

```
BUInt32 BoapMcComms::setTimeout (
            BUInt32 timeoutUs )
```

Sets the call timeout returning the current value.

7.58.2.7 processRx()

```
BError BoapMcComms::processRx (
            BTimeout timeoutUs = BTimeoutForever )  [virtual]
```

Process any RX packets queuing them as needed.

!!! This should wait on comms for timeoutUs !!!

7.58.2.8 processRequests()

```
BError BoapMcComms::processRequests (
            BTimeout timeoutUs = BTimeoutForever )  [virtual]
```

Check and process all requests.

7.58.2.9 processRequest()

```
BError BoapMcComms::processRequest (
            BTimeout timeoutUs = BTimeoutForever )  [virtual]
```

Check and process any request.

7.58.2.10 processPacket()

```
BError BoapMcComms::processPacket (
            BoapMcPacket & rx,
            BoapMcPacket & tx )  [virtual]
```

Process a recieved packet.

7.58.2.11 performCall()

```
BError BoapMcComms::performCall ( )  [protected]
```

Performs a RPC call to the remote side.

7.58.2.12 performSend()

```
BError BoapMcComms::performSend ( )  [protected]
```

Performs a RPC send to the remote side.

7.58.2.13 packetSend()

```
BError BoapMcComms::packetSend (
            BoapMcPacket & packet )  [protected]
```

Receives a packet.

7.58.2.14 packetRecv()

```
BError BoapMcComms::packetRecv (
            BoapMcPacket & packet )  [protected]
```

Receives a packet.

7.58.3 Member Data Documentation

7.58.3.1 othreaded

```
Bool BoapMcComms::othreaded  [protected]
```

7.58.3.2 olockCall

```
BMutex BoapMcComms::olockCall  [protected]
```

Lock for RPC calls. Only one at a time.

7.58.3.3 olockTx

```
BMutex BoapMcComms::olockTx  [protected]
```

Lock for TX.

7.58.3.4 ocomms

```
BComms* BoapMcComms::ocomms  [protected]
```

7.58.3.5 oapiVersion

```
BUInt32 BoapMcComms::oapiVersion  [protected]
```

7.58.3.6 oslave

```
Bool BoapMcComms::oslave  [protected]
```

Set slave mode.

7.58.3.7 otimeout

```
BUInt32 BoapMcComms::otimeout  [protected]
```

The timeout in us for calls.

7.58.3.8 oaddressTo

```
BUInt8 BoapMcComms::oaddressTo  [protected]
```

7.58.3.9 oaddressFrom

```
BUInt8 BoapMcComms::oaddressFrom  [protected]
```

7.58.3.10 opacket

BoapMcPacket BoapMcComms::opacket [protected]

Packet RX buffer.

7.58.3.11 opacketTx

BoapMcPacket BoapMcComms::opacketTx [protected]

Packet TX buffer for calls.

7.58.3.12 opacketRx

BoapMcPacket BoapMcComms::opacketRx [protected]

Packet RX buffer for calls.

7.58.3.13 opacketRxSema

BSemaphore BoapMcComms::opacketRxSema [protected]

Wait RX semaphore.

7.58.3.14 opacketReqTx

BoapMcPacket BoapMcComms::opacketReqTx [protected]

Packet TX buffer for requests.

7.58.3.15 opacketReqRx

BoapMcPacket BoapMcComms::opacketReqRx [protected]

Packet RX buffer for requests.

7.58.3.16 opacketReqQueue

```
BQueue<BoapMcPacket> BoapMcComms::opacketReqQueue  [protected]
```

Packet RX buffer queue for requests.

7.58.3.17 opacketTxQueue

```
BFifo<BoapMcPacket> BoapMcComms::opacketTxQueue  [protected]
```

Packet TX Queue.

7.58.3.18 opacketTxQueueWriteNum

```
BSemaphoreCount BoapMcComms::opacketTxQueueWriteNum  [protected]
```

Packet TX Queue number.

7.58.3.19 opacketTxSema

```
BSemaphore BoapMcComms::opacketTxSema  [protected]
```

Wait for TX semaphore.

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

- BoapMc.h
- BoapMc.cpp

7.59 BoapMcPacket Class Reference

```
#include <BoapMc.h>
```

Public Attributes

- BoapMcPacketHead head
- char data [256 - sizeof(BoapMcPacketHead)]

7.59.1 Member Data Documentation

7.59.1.1 head

```
BoapMcPacketHead BoapMcPacket::head
```

7.59.1.2 data

```
char BoapMcPacket::data[256 - sizeof(BoapMcPacketHead)]
```

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

- BoapMc.h

7.60 BoapMcPacketHead Struct Reference

```
#include <BoapMc.h>
```

Public Attributes

- BUInt8 length
- BUInt8 addressTo
- BUInt8 addressFrom
- BUInt8 cmd
- BUInt16 error
- BUInt16 checksum

7.60.1 Member Data Documentation

7.60.1.1 length

```
BUInt8 BoapMcPacketHead::length
```

7.60.1.2 addressTo

```
BUInt8 BoapMcPacketHead::addressTo
```

7.60.1.3 addressFrom

```
BUInt8 BoapMcPacketHead::addressFrom
```

7.60.1.4 cmd

```
BUInt8 BoapMcPacketHead::cmd
```

7.60.1.5 error

```
BUInt16 BoapMcPacketHead::error
```

7.60.1.6 checksum

```
BUInt16 BoapMcPacketHead::checksum
```

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

- BoapMc.h

7.61 BoapMcServiceObject Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcServiceObject ()
- virtual ~BoapMcServiceObject ()
- virtual BError process (BoapMcPacket &rx, BoapMcPacket &tx)
- virtual BError processEvent (BoapMcPacket &rx)

Protected Member Functions

- BError sendEvent (BoapMcPacket &tx)

Protected Attributes

- BUInt32 oapiVersion

7.61.1 Constructor & Destructor Documentation

7.61.1.1 BoapMcServiceObject()

```
BoapMcServiceObject::BoapMcServiceObject ( )
```

7.61.1.2 ~BoapMcServiceObject()

```
BoapMcServiceObject::~BoapMcServiceObject ( )  [virtual]
```

7.61.2 Member Function Documentation

7.61.2.1 process()

```
BError BoapMcServiceObject::process (
            BoapMcPacket & rx,
            BoapMcPacket & tx )  [virtual]
```

7.61.2.2 processEvent()

```
BError BoapMcServiceObject::processEvent (
            BoapMcPacket & rx )  [virtual]
```

7.61.2.3 sendEvent()

```
BError BoapMcServiceObject::sendEvent (
            BoapMcPacket & tx )  [protected]
```

7.61.3 Member Data Documentation

7.61.3.1 oapiVersion

```
BUInt32 BoapMcServiceObject::oapiVersion  [protected]
```

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

- BoapMc.h
- BoapMc.cpp

7.62 BoapMcSignalObject Class Reference

```
#include <BoapMc.h>
```

Public Member Functions

- BoapMcSignalObject (BComms &comms)

Protected Member Functions

- BError performSend (BoapMcPacket &tx)

Protected Attributes

- BComms & ocomms

7.62.1 Constructor & Destructor Documentation

7.62.1.1 BoapMcSignalObject()

```
BoapMcSignalObject::BoapMcSignalObject (
            BComms & comms )
```

7.62.2 Member Function Documentation

7.62.2.1 performSend()

```
BError BoapMcSignalObject::performSend (
            BoapMcPacket & tx )  [protected]
```

7.62.3 Member Data Documentation

7.62.3.1 ocomms

```
BComms& BoapMcSignalObject::ocomms  [protected]
```

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

- BoapMc.h
- BoapMc.cpp

7.63 Boapns::BoapEntry Class Reference

```
#include <BoapnsD.h>
```

Public Member Functions

- BoapEntry (BString name=BString(), BString hostName=BString(), BList< BString > addressList=BList< BString >(), BUInt32 port=0, BUInt32 service=0)

Public Attributes

- BString name
- BString hostName
- BList< BString > addressList
- BUInt32 port
- BUInt32 service

7.63.1 Constructor & Destructor Documentation

7.63.1.1 BoapEntry()

```
Boapns::BoapEntry::BoapEntry (
            BString name = BString(),
            BString hostName = BString(),
            BList< BString > addressList = BList<BString >(),
            BUInt32 port = 0,
            BUInt32 service = 0 )
```

7.63.2 Member Data Documentation

7.63.2.1 name

```
BString Boapns::BoapEntry::name
```

7.63.2.2 hostName

```
BString Boapns::BoapEntry::hostName
```

7.63.2.3 addressList

```
BList<BString > Boapns::BoapEntry::addressList
```

7.63.2.4 port

```
BUInt32 Boapns::BoapEntry::port
```

7.63.2.5 service

```
BUInt32 Boapns::BoapEntry::service
```

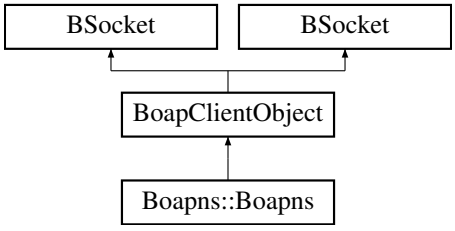
The documentation for this class was generated from the following files:

- BoapnsD.h
- BoapnsD.cpp

7.64 Boapns::Boapns Class Reference

```
#include <BoapnsC.h>
```

Inheritance diagram for Boapns::Boapns:



Public Member Functions

- Boapns (BString name="")
- BError getVersion (BString &version)

  Get the Boapns version.
- BError getEntryList (BList< BoapEntry > &entryList)
- BError getEntry (BString name, BoapEntry &entry)
- BError addEntry (BoapEntry entry)
- BError delEntry (BString name)
- BError getNewName (BString &name)

Additional Inherited Members

7.64.1 Constructor & Destructor Documentation

7.64.1.1 Boapns()

```
Boapns::Boapns::Boapns (
            BString name = "" )
```

7.64.2 Member Function Documentation

7.64.2.1 getVersion()

```
BError Boapns::Boapns::getVersion (
            BString & version )
```

Get the Boapns version.

7.64.2.2 getEntryList()

```
BError Boapns::Boapns::getEntryList (
            BList< BoapEntry > & entryList )
```

7.64.2.3 getEntry()

```
BError Boapns::Boapns::getEntry (
            BString name,
            BoapEntry & entry )
```

7.64.2.4 addEntry()

```
BError Boapns::Boapns::addEntry (
            BoapEntry entry )
```

7.64.2.5 delEntry()

```
BError Boapns::Boapns::delEntry (
            BString name )
```

7.64.2.6 getNewName()

```
BError Boapns::Boapns::getNewName (
            BString & name )
```

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

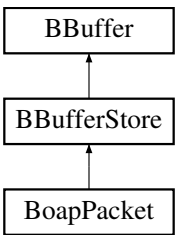
- BoapnsC.h
- BoapnsC.cpp

7.65 BoapPacket Class Reference

Boap packet.

```
#include <Boap.h>
```

Inheritance diagram for BoapPacket:



Public Member Functions

- BoapPacket ()
- ~BoapPacket ()
- BUInt32 getCmd ()
- int peekHead (BoapPacketHead &head)
- int pushHead (BoapPacketHead &head)
- int popHead (BoapPacketHead &head)
- void updateHead ()
- BoapPacket ()
- ~BoapPacket ()
- int resize (int size)
- BError setData (void *data, int nbytes)
- int nbytes ()
- char * data ()
- int pushHead (BoapPacketHead &head)
- int push (Int8 v)
- int push (UInt8 v)
- int push (Int16 v)
- int push (UInt16 v)
- int push (Int32 v)
- int push (UInt32 v)
- int push (BString &v)
- int push (Double v)
- int push (BError &v)
- int push (UInt32 nBytes, const void *data)
- int popHead (BoapPacketHead &head)
- int pop (Int8 &v)
- int pop (UInt8 &v)
- int pop (Int16 &v)
- int pop (UInt16 &v)
- int pop (Int32 &v)
- int pop (UInt32 &v)
- int pop (BString &v)
- int pop (Double &v)
- int pop (BError &v)
- int pop (UInt32 nBytes, void *data)

Additional Inherited Members

7.65.1 Detailed Description

Boap packet.

7.65.2 Constructor & Destructor Documentation

7.65.2.1 BoapPacket() [1/2]

```
BoapPacket::BoapPacket ( )
```

7.65.2.2 ~BoapPacket() [1/2]

```
BoapPacket::~BoapPacket ( )
```

7.65.2.3 BoapPacket() [2/2]

```
BoapPacket::BoapPacket ( )
```

7.65.2.4 ~BoapPacket() [2/2]

```
BoapPacket::~BoapPacket ( )
```

7.65.3 Member Function Documentation

7.65.3.1 getCmd()

```
BUInt32 BoapPacket::getCmd ( )
```

7.65.3.2 peekHead()

```
int BoapPacket::peekHead (
            BoapPacketHead & head )
```

7.65.3.3 pushHead() [1/2]

```
int BoapPacket::pushHead (
            BoapPacketHead & head )
```

7.65.3.4 popHead() [1/2]

```
int BoapPacket::popHead (
            BoapPacketHead & head )
```

7.65.3.5 updateHead()

```
void BoapPacket::updateHead ( )
```

7.65.3.6 resize()

```
int BoapPacket::resize (
            int size )
```

7.65.3.7 setData()

```
BError BoapPacket::setData (
            void * data,
            int nbytes )
```

7.65.3.8 nbytes()

```
int BoapPacket::nbytes ( )
```

7.65.3.9 data()

```
char * BoapPacket::data ( )
```

7.65.3.10 pushHead() [2/2]

```
int BoapPacket::pushHead (
            BoapPacketHead & head )
```

7.65.3.11 push() [1/10]

```
int BoapPacket::push (
            Int8 v )
```

7.65.3.12 push() [2/10]

```
int BoapPacket::push (
            UInt8 v )
```

7.65.3.13 push() [3/10]

```
int BoapPacket::push (
            Int16 v )
```

7.65.3.14 push() [4/10]

```
int BoapPacket::push (
            UInt16 v )
```

7.65.3.15 push() [5/10]

```
int BoapPacket::push (
            Int32 v )
```

7.65.3.16 push() [6/10]

```
int BoapPacket::push (
            UInt32 v )
```

7.65.3.17 push() [7/10]

```
int BoapPacket::push (
            BString & v )
```

7.65.3.18 push() [8/10]

```
int BoapPacket::push (
            Double v )
```

7.65.3.19 push() [9/10]

```
int BoapPacket::push (
            BError & v )
```

7.65.3.20 push() [10/10]

```
int BoapPacket::push (
            UInt32 nBytes,
            const void * data )
```

7.65.3.21 popHead() [2/2]

```
int BoapPacket::popHead (
            BoapPacketHead & head )
```

7.65.3.22 pop() [1/10]

```
int BoapPacket::pop (
            Int8 & v )
```

7.65.3.23 pop() [2/10]

```
int BoapPacket::pop (
            UInt8 & v )
```

7.65.3.24 pop() [3/10]

```
int BoapPacket::pop (
            Int16 & v )
```

7.65.3.25 pop() [4/10]

```
int BoapPacket::pop (
            UInt16 & v )
```

7.65.3.26 pop() [5/10]

```
int BoapPacket::pop (
            Int32 & v )
```

7.65.3.27 pop() [6/10]

```
int BoapPacket::pop (
            UInt32 & v )
```

7.65.3.28 pop() [7/10]

```
int BoapPacket::pop (
            BString & v )
```

7.65.3.29 pop() [8/10]

```
int BoapPacket::pop (
            Double & v )
```

7.65.3.30 pop() [9/10]

```
int BoapPacket::pop (
            BError & v )
```

7.65.3.31 pop() [10/10]

```
int BoapPacket::pop (
            UInt32 nBytes,
            void * data )
```

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

- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.66 BoapPacketHead Struct Reference

Boap packet header.

`#include <Boap.h>`

Public Attributes

- BUInt32 type
- BUInt32 length
- BUInt32 service
- BUInt32 cmd
- UInt32 length
- BoapType type
- BoapService service
- UInt32 cmd
- UInt32 reserved [12]

7.66.1 Detailed Description

Boap packet header.

7.66.2 Member Data Documentation

7.66.2.1 type [1/2]

```
BUInt32 BoapPacketHead::type
```

7.66.2.2 length [1/2]

```
BUInt32 BoapPacketHead::length
```

7.66.2.3 service [1/2]

```
BUInt32 BoapPacketHead::service
```

7.66.2.4 cmd [1/2]

```
BUInt32 BoapPacketHead::cmd
```

7.66.2.5 length [2/2]

```
UInt32 BoapPacketHead::length
```

7.66.2.6 type [2/2]

```
BoapType BoapPacketHead::type
```

7.66.2.7 service [2/2]

```
BoapService BoapPacketHead::service
```

7.66.2.8 cmd [2/2]

```
UInt32 BoapPacketHead::cmd
```

7.66.2.9 reserved

```
UInt32 BoapPacketHead::reserved[12]
```

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

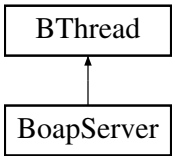
- Boap.h
- BoapSimple.h

7.67 BoapServer Class Reference

Boap server.

```
#include <Boap.h>
```

Inheritance diagram for BoapServer:



Public Types

- enum { NOTHREADS =0 , THREADED =1 }

Public Member Functions

- BoapServer ()
- virtual ~BoapServer ()
- virtual BError init (BString boapNsHost="", int port=0, int threaded=0, int isBoapns=0)
- virtual BError run (int inThread=0)
- virtual BError process (BoapServerConnection *conn, BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)
- virtual BError addObject (BoapServiceObject *object)
- virtual BError sendEvent (BoapPacket &tx)
- virtual BError processEvent (int fd)
- virtual void clientGone (BoapServerConnection *client)
- BSocket & getSocket ()
- BSocket & getEventSocket ()
- BString getHostName ()
- int getConnectionsNumber ()
- void closeConnections ()
- virtual BoapServerConnection * newConnection (int fd, BSocketAddressINET address)
- BoapServer ()
- BError init (int boapNs=0)
- BError run ()
- BError processEvent (BoapPacket &rx)
- BError addObject (BoapServiceObject *object)
- BError process (int fd)
- BError sendEvent (BoapPacket &tx)
- BSocket & getSocket ()
- BSocket & getEventSocket ()
- BError processEvent (int fd)
- BString getHostName ()

Public Attributes

- BUInt64 onumOperations

Protected Member Functions

- void * function ()

Protected Attributes

- BMutex olock
- int othreaded
- int oisBoapns
- Boapns::Boapns * oboapns
- BList< BoapServerConnection * > oclients
- BEvent1Int oclientGoneEvent
- BList< BoapServiceEntry > oservices
- BPoll opoll
- BSocket onet
- BSocket onetEvent
- BSocketAddressINET onetEventAddress
- BString ohostName

7.67.1 Detailed Description

Boap server.

7.67.2 Member Enumeration Documentation

7.67.2.1 anonymous enum

```
anonymous enum
```

Enumerator

NOTHREADS	
THREADED	

7.67.3 Constructor & Destructor Documentation

7.67.3.1 BoapServer() [1/2]

```
BoapServer::BoapServer ( )
```

7.67.3.2 ~BoapServer()

```
BoapServer::~BoapServer ( )  [virtual]
```

7.67.3.3 BoapServer() [2/2]

```
BoapServer::BoapServer ( )
```

7.67.4 Member Function Documentation

7.67.4.1 init() [1/2]

```
BError BoapServer::init (
            BString boapNsHost = "",
            int port = 0,
            int threaded = 0,
            int isBoapns = 0 )  [virtual]
```

7.67.4.2 run() [1/2]

```
BError BoapServer::run (
            int inThread = 0 )  [virtual]
```

7.67.4.3 process() [1/2]

```
BError BoapServer::process (
            BoapServerConnection * conn,
            BoapPacket & rx,
            BoapPacket & tx )  [virtual]
```

7.67.4.4 processEvent() [1/4]

```
BError BoapServer::processEvent (
            BoapPacket & rx )  [virtual]
```

7.67.4.5 addObject() [1/2]

```
BError BoapServer::addObject (
            BoapServiceObject * object )  [virtual]
```

7.67.4.6 sendEvent() [1/2]

```
BError BoapServer::sendEvent (
            BoapPacket & tx )  [virtual]
```

7.67.4.7 processEvent() [2/4]

```
BError BoapServer::processEvent (
            int fd )  [virtual]
```

7.67.4.8 clientGone()

```
void BoapServer::clientGone (
            BoapServerConnection * client )  [virtual]
```

7.67.4.9 getSocket() [1/2]

```
BSocket & BoapServer::getSocket ( )
```

7.67.4.10 getEventSocket() [1/2]

```
BSocket & BoapServer::getEventSocket ( )
```

7.67.4.11 getHostName() [1/2]

```
BString BoapServer::getHostName ( )
```

7.67.4.12 getConnectionsNumber()

```
int BoapServer::getConnectionsNumber ( )
```

7.67.4.13 closeConnections()

```
void BoapServer::closeConnections ( )
```

7.67.4.14 newConnection()

```
BoapServerConnection * BoapServer::newConnection (
            int fd,
            BSocketAddressINET address )  [virtual]
```

7.67.4.15 function()

```
void * BoapServer::function ( )  [protected], [virtual]
```

Reimplemented from BThread.

7.67.4.16 init() [2/2]

```
BError BoapServer::init (
            int boapNs = 0 )
```

7.67.4.17 run() [2/2]

```
BError BoapServer::run ( )
```

7.67.4.18 processEvent() [3/4]

```
BError BoapServer::processEvent (
            BoapPacket & rx )
```

7.67.4.19 addObject() [2/2]

```
BError BoapServer::addObject (
            BoapServiceObject * object )
```

7.67.4.20 process() [2/2]

```
BError BoapServer::process (
            int fd )
```

7.67.4.21 sendEvent() [2/2]

```
BError BoapServer::sendEvent (
            BoapPacket & tx )
```

7.67.4.22 getSocket() [2/2]

```
BSocket& BoapServer::getSocket ( )
```

7.67.4.23 getEventSocket() [2/2]

```
BSocket& BoapServer::getEventSocket ( )
```

7.67.4.24 processEvent() [4/4]

```
BError BoapServer::processEvent (
            int fd )
```

7.67.4.25 getHostName() [2/2]

```
BString BoapServer::getHostName ( )
```

7.67.5 Member Data Documentation

7.67.5.1 olock

```
BMutex BoapServer::olock  [protected]
```

7.67.5.2 othreaded

```
int BoapServer::othreaded  [protected]
```

7.67.5.3 oisBoapns

```
int BoapServer::oisBoapns  [protected]
```

7.67.5.4 oboapns

```
Boapns::Boapns* BoapServer::oboapns  [protected]
```

7.67.5.5 oclients

```
BList<BoapServerConnection*> BoapServer::oclients  [protected]
```

7.67.5.6 oclientGoneEvent

```
BEvent1Int BoapServer::oclientGoneEvent  [protected]
```

7.67.5.7 oservices

```
BList< BoapServiceEntry > BoapServer::oservices  [protected]
```

7.67.5.8 opoll

```
BPoll BoapServer::opoll  [protected]
```

7.67.5.9 onet

```
BSocket BoapServer::onet  [protected]
```

7.67.5.10 onetEvent

```
BSocket BoapServer::onetEvent  [protected]
```

7.67.5.11 onetEventAddress

```
BSocketAddressINET BoapServer::onetEventAddress  [protected]
```

7.67.5.12 ohostName

```
BString BoapServer::ohostName  [protected]
```

7.67.5.13 onumOperations

```
BUInt64 BoapServer::onumOperations
```

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

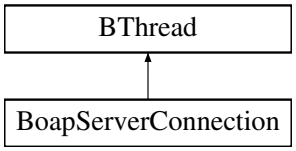
- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.68 BoapServerConnection Class Reference

Boap server connection.

`#include <Boap.h>`

Inheritance diagram for BoapServerConnection:



Public Member Functions

- BoapServerConnection (BoapServer &boapServer, int fd)
- virtual ~BoapServerConnection ()
- virtual BError init ()

  Initialise connection.
- virtual BError process ()
- virtual BSocket & getSocket ()
- virtual void setMaxLength (BUInt32 maxLength)
- virtual BError getHead (BoapPacketHead &head)
- virtual BError validate ()

  Validate the connection.

7.68.1 Detailed Description

Boap server connection.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 BoapServerConnection()

```
BoapServerConnection::BoapServerConnection (
            BoapServer & boapServer,
            int fd )
```

7.68.2.2 ~BoapServerConnection()

```
BoapServerConnection::~BoapServerConnection ( )  [virtual]
```

7.68.3 Member Function Documentation

7.68.3.1 init()

```
BError BoapServerConnection::init ( )  [virtual]
```

Initialise connection.

7.68.3.2 process()

```
BError BoapServerConnection::process ( )  [virtual]
```

7.68.3.3 getSocket()

```
BSocket & BoapServerConnection::getSocket ( )  [virtual]
```

7.68.3.4 setMaxLength()

```
void BoapServerConnection::setMaxLength (
            BUInt32 maxLength )  [virtual]
```

7.68.3.5 getHead()

```
BError BoapServerConnection::getHead (
            BoapPacketHead & head )  [virtual]
```

7.68.3.6 validate()

```
BError BoapServerConnection::validate ( )  [virtual]
```

Validate the connection.

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

- Boap.h
- Boap.cpp

7.69 BoapServiceEntry Class Reference

Boap server single service entry.

`#include <Boap.h>`

Public Member Functions

- BoapServiceEntry (BoapService service=0, BoapServiceObject *object=0)
- BoapServiceEntry (BoapService service=0, BoapServiceObject *object=0)

Public Attributes

- BoapService oservice
- BoapServiceObject * oobject

7.69.1 Detailed Description

Boap server single service entry.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 BoapServiceEntry() [1/2]

```
BoapServiceEntry::BoapServiceEntry (
            BoapService service = 0,
            BoapServiceObject * object = 0 )  [inline]
```

7.69.2.2 BoapServiceEntry() [2/2]

```
BoapServiceEntry::BoapServiceEntry (
            BoapService service = 0,
            BoapServiceObject * object = 0 )  [inline]
```

7.69.3 Member Data Documentation

7.69.3.1 oservice

```
BoapService BoapServiceEntry::oservice
```

7.69.3.2 oobject

```
BoapServiceObject * BoapServiceEntry::oobject
```

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

- Boap.h
- BoapSimple.h

7.70 BoapServiceObject Class Reference

Boap service object.

```
#include <Boap.h>
```

Public Member Functions

- BoapServiceObject (BoapServer &server, BString name="")
- virtual ~BoapServiceObject ()
- BError setName (BString name)
- BError sendEvent (BString signalName, BInt32 arg)
- virtual BError processEvent (BString objectName, BString name, BInt32 arg)
- BString name ()
- BUInt32 apiVersion ()

  Returns the API version.
- BError doPing (BoapServerConnection *conn, BoapPacket &rx, BoapPacket &tx)
- BError doConnectionPriority (BoapServerConnection *conn, BoapPacket &rx, BoapPacket &tx)
- BError process (BoapServerConnection *conn, BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)
- BoapServiceObject (BoapServer &server, BString name)
- virtual ~BoapServiceObject ()
- BError sendEvent (BString signalName, Int32 arg)
- virtual BError processEvent (BString objectName, BString name, Int32 arg)
- BString name ()
- BError process (BoapPacket &rx, BoapPacket &tx)
- virtual BError processEvent (BoapPacket &rx)

Protected Member Functions

- BError sendEvent (BoapPacket &tx)
- BError sendEvent (BoapPacket &tx)

Protected Attributes

- BoapServer & oserver
- BString oname
- BUInt32 oapiVersion
- BList< BoapFuncEntry > ofuncList

7.70.1 Detailed Description

Boap service object.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 BoapServiceObject() [1/2]

```
BoapServiceObject::BoapServiceObject (
            BoapServer & server,
            BString name = "" )
```

7.70.2.2 ~BoapServiceObject() [1/2]

```
BoapServiceObject::~BoapServiceObject ( )  [virtual]
```

7.70.2.3 BoapServiceObject() [2/2]

```
BoapServiceObject::BoapServiceObject (
            BoapServer & server,
            BString name )
```

7.70.2.4 ~BoapServiceObject() [2/2]

```
virtual BoapServiceObject::~BoapServiceObject ( )  [virtual]
```

7.70.3 Member Function Documentation

7.70.3.1 setName()

```
BError BoapServiceObject::setName (
            BString name )
```

7.70.3.2 sendEvent() [1/4]

```
BError BoapServiceObject::sendEvent (
            BString signalName,
            BInt32 arg )
```

7.70.3.3 processEvent() [1/4]

```
BError BoapServiceObject::processEvent (
            BString objectName,
            BString name,
            BInt32 arg )  [virtual]
```

7.70.3.4 name() [1/2]

```
BString BoapServiceObject::name ( )
```

7.70.3.5 apiVersion()

```
BUInt32 BoapServiceObject::apiVersion ( )
```

Returns the API version.

7.70.3.6 doPing()

```
BError BoapServiceObject::doPing (
            BoapServerConnection * conn,
            BoapPacket & rx,
            BoapPacket & tx )
```

7.70.3.7 doConnectionPriority()

```
BError BoapServiceObject::doConnectionPriority (
            BoapServerConnection * conn,
            BoapPacket & rx,
            BoapPacket & tx )
```

7.70.3.8 process() [1/2]

```
BError BoapServiceObject::process (
            BoapServerConnection * conn,
            BoapPacket & rx,
            BoapPacket & tx )
```

7.70.3.9 processEvent() [2/4]

```
BError BoapServiceObject::processEvent (
            BoapPacket & rx )  [virtual]
```

7.70.3.10 sendEvent() [2/4]

```
BError BoapServiceObject::sendEvent (
            BoapPacket & tx )  [protected]
```

7.70.3.11 sendEvent() [3/4]

```
BError BoapServiceObject::sendEvent (
            BString signalName,
            Int32 arg )
```

7.70.3.12 processEvent() [3/4]

```
virtual BError BoapServiceObject::processEvent (
            BString objectName,
            BString name,
            Int32 arg )  [virtual]
```

7.70.3.13 name() [2/2]

```
BString BoapServiceObject::name ( )
```

7.70.3.14 process() [2/2]

```
BError BoapServiceObject::process (
            BoapPacket & rx,
            BoapPacket & tx )
```

7.70.3.15 processEvent() [4/4]

```
virtual BError BoapServiceObject::processEvent (
            BoapPacket & rx )  [virtual]
```

7.70.3.16 sendEvent() [4/4]

```
BError BoapServiceObject::sendEvent (
            BoapPacket & tx )  [protected]
```

7.70.4 Member Data Documentation

7.70.4.1 oserver

```
BoapServer & BoapServiceObject::oserver  [protected]
```

7.70.4.2 oname

```
BString BoapServiceObject::oname  [protected]
```

7.70.4.3 oapiVersion

```
BUInt32 BoapServiceObject::oapiVersion  [protected]
```

7.70.4.4 ofuncList

```
BList< BoapFuncEntry > BoapServiceObject::ofuncList  [protected]
```

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

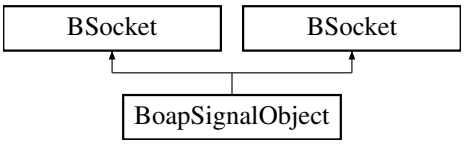
- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.71 BoapSignalObject Class Reference

A Boap object to send signals using an RPC mechanism.

```
#include <Boap.h>
```

Inheritance diagram for BoapSignalObject:



Public Member Functions

- BoapSignalObject ()
- BoapSignalObject ()

Protected Member Functions

- BError performSend (BoapPacket &tx)
- BError performSend (BoapPacket &tx)

Protected Attributes

- BoapPacket otx
- BoapPacket orx

Additional Inherited Members

7.71.1 Detailed Description

A Boap object to send signals using an RPC mechanism.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 BoapSignalObject() [1/2]

```
BoapSignalObject::BoapSignalObject ( )
```

7.71.2.2 BoapSignalObject() [2/2]

```
BoapSignalObject::BoapSignalObject ( )
```

7.71.3 Member Function Documentation

7.71.3.1 performSend() [1/2]

```
BError BoapSignalObject::performSend (
            BoapPacket & tx )  [protected]
```

7.71.3.2 performSend() [2/2]

```
BError BoapSignalObject::performSend (
            BoapPacket & tx )  [protected]
```

7.71.4 Member Data Documentation

7.71.4.1 otx

```
BoapPacket BoapSignalObject::otx  [protected]
```

7.71.4.2 orx

```
BoapPacket BoapSignalObject::orx  [protected]
```

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

- Boap.h
- BoapSimple.h
- Boap.cpp
- BoapSimple.cc

7.72 BObj Class Reference

A generic object base class that has runtime definable data feilds.

```
#include <BObj.h>
```

Public Member Functions

- BObj ()
- virtual ~BObj ()
- virtual const char * getType () const
- virtual const BObjMember * getMembers () const
- virtual BError getMembers (BDictString &members)
- virtual BError getMember (BString name, BString &value)
- virtual BError setMembers (BDictString &members)
- virtual BError setMember (BString name, BString value)
- virtual void membersPrint () const

  Prints out members.
- virtual BString getDebugString ()

  Returns contents as a debug string.

7.72.1 Detailed Description

A generic object base class that has runtime definable data feilds.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 BObj()

```
BObj::BObj ( )
```

7.72.2.2 ~BObj()

```
BObj::~BObj ( )  [virtual]
```

7.72.3 Member Function Documentation

7.72.3.1 getType()

```
const char * BObj::getType ( ) const  [virtual]
```

7.72.3.2 getMembers() [1/2]

```
const BObjMember * BObj::getMembers ( ) const  [virtual]
```

7.72.3.3 getMembers() [2/2]

```
BError BObj::getMembers (
            BDictString & members )  [virtual]
```

7.72.3.4 getMember()

```
BError BObj::getMember (
            BString name,
            BString & value )  [virtual]
```

7.72.3.5 setMembers()

```
BError BObj::setMembers (
            BDictString & members )  [virtual]
```

7.72.3.6 setMember()

```
BError BObj::setMember (
            BString name,
            BString value )  [virtual]
```

7.72.3.7 membersPrint()

```
void BObj::membersPrint ( ) const  [virtual]
```

Prints out members.

7.72.3.8 getDebugString()

```
BString BObj::getDebugString ( )  [virtual]
```

Returns contents as a debug string.

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

- BObj.h
- BObj.cpp

7.73 BObjMember Struct Reference

A structure to define a member of a generic BObj.

```
#include <BTypes.h>
```

Public Attributes

- BType type
- BTypeComp typeComp
- BUInt16 dataOffset
- BUInt16 size
- const char * typeName
- const char * name

7.73.1 Detailed Description

A structure to define a member of a generic BObj.

7.73.2 Member Data Documentation

7.73.2.1 type

```
BType BObjMember::type
```

7.73.2.2 typeComp

```
BTypeComp BObjMember::typeComp
```

7.73.2.3 dataOffset

```
BUInt16 BObjMember::dataOffset
```

7.73.2.4 size

```
BUInt16 BObjMember::size
```

7.73.2.5 typeName

```
const char* BObjMember::typeName
```

7.73.2.6 name

```
const char* BObjMember::name
```

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

- BTypes.h

7.74 BPoll Class Reference

This class provides an interface for polling a number of file descriptors. It uses round robin polling.

`#include <BPoll.h>`

Public Types

- typedef struct pollfd PollFd

Public Member Functions

- BPoll ()
- ~BPoll ()
- void append (int fd, int events=POLLIN|POLLERR|POLLHUP|POLLNVAL)

  Append a file descriptor to polling list.
- void delFd (int fd)

  Remove a file descriptor from polling list.
- BError doPoll (int &fd, int timeoutUs=-1)

  Perform polling operation.
- BError doPollEvents (int &fd, int &events, int timeoutUs=-1)

  Perform polling operation and return events.
- int getPollFdsNum ()
- PollFd * getPollFds ()
- void clear ()

7.74.1 Detailed Description

This class provides an interface for polling a number of file descriptors. It uses round robin polling.

7.74.2 Member Typedef Documentation

7.74.2.1 PollFd

`typedef struct pollfd BPoll::PollFd`

7.74.3 Constructor & Destructor Documentation

7.74.3.1 BPoll()

```
BPoll::BPoll ( )
```

7.74.3.2 ~BPoll()

```
BPoll::~BPoll ( )
```

7.74.4 Member Function Documentation

7.74.4.1 append()

```
void BPoll::append (
            int fd,
            int events = POLLIN|POLLERR|POLLHUP|POLLNVAL )
```

Append a file descriptor to polling list.

7.74.4.2 delFd()

```
void BPoll::delFd (
            int fd )
```

Remove a file descriptor from polling list.

7.74.4.3 doPoll()

```
BError BPoll::doPoll (
            int & fd,
            int timeoutUs = -1 )
```

Perform polling operation.

7.74.4.4 doPollEvents()

```
BError BPoll::doPollEvents (
            int & fd,
            int & events,
            int timeoutUs = -1 )
```

Perform polling operation and return events.

7.74.4.5 getPollFdsNum()

```
int BPoll::getPollFdsNum ( )
```

7.74.4.6 getPollFds()

```
BPoll::PollFd * BPoll::getPollFds ( )
```

7.74.4.7 clear()

```
void BPoll::clear ( )
```

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

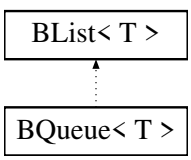
- BPoll.h
- BPoll.cpp

7.75 BQueue< T > Class Template Reference

Provides a thread save queue of objects that can be used to communicate between threads.

`#include <BQueue.h>`

Inheritance diagram for BQueue< T >:



Public Member Functions

- BQueue (BUInt size)
- ~BQueue ()
- void clear ()

  Clear the queue.
- BUInt writeAvailable () const
- BError write (const T &v, BTimeout timeout=BTimeoutForever)

  Append an item onto the queue.
- BUInt readAvailable () const
- BError read (T &v, BTimeout timeout=BTimeoutForever)

  Get an item from the queue.

7.75.1 Detailed Description

template<class T>
class BQueue< T >

Provides a thread save queue of objects that can be used to communicate between threads.

7.75.2 Constructor & Destructor Documentation

7.75.2.1 BQueue()

```
template<class T >
BQueue< T >::BQueue (
            BUInt size )
```

7.75.2.2 ~BQueue()

```
template<class T >
BQueue< T >::~BQueue
```

7.75.3 Member Function Documentation

7.75.3.1 clear()

```
template<class T >
void BQueue< T >::clear  [virtual]
```

Clear the queue.

Reimplemented from BList< T >.

7.75.3.2 writeAvailable()

```
template<class T >
BUInt BQueue< T >::writeAvailable
```

7.75.3.3 write()

```
template<class T >
BError BQueue< T >::write (
            const T & v,
            BTimeout timeout = BTimeoutForever )
```

Append an item onto the queue.

7.75.3.4 readAvailable()

```
template<class T >
BUInt BQueue< T >::readAvailable
```

7.75.3.5 read()

```
template<class T >
BError BQueue< T >::read (
            T & v,
            BTimeout timeout = BTimeoutForever )
```

Get an item from the queue.

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

- BQueue.h

7.76 BRefData Class Reference

A pointer to a variable sized data area with reference counting so the data areas can be shared.

`#include <BRefData.h>`

Public Member Functions

- BRefData ()
- BRefData (int len)
- BRefData (const BRefData &refData)
- ~BRefData ()
- BRefData * copy ()

  Create a copy of this reference for writing, if necessary

- BRefData * addRef ()

  Increment the reference counter.
- int deleteRef ()

  Decrement the reference counter.
- char * data ()

  Return the raw data pointer.
- int len ()

  Return the length in bytes.
- BRefData & operator= (const BRefData &refData)
- void setLen (int len)

  Set the length in bytes. Note should only be used if orefCount = 1.

7.76.1 Detailed Description

A pointer to a variable sized data area with reference counting so the data areas can be shared.

This is Thread safe to a degree. The reference counting is protected. However, setLen() is not and should be protected at a higher level.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 BRefData() [1/3]

`BRefData::BRefData ( )`

7.76.2.2 BRefData() [2/3]

```
BRefData::BRefData (
            int len )
```

7.76.2.3 BRefData() [3/3]

```
BRefData::BRefData (
            const BRefData & refData )
```

7.76.2.4 ~BRefData()

```
BRefData::~BRefData ( )
```

7.76.3 Member Function Documentation

7.76.3.1 copy()

```
BRefData * BRefData::copy ( )
```

Create a copy of this reference for writing, if necessary

7.76.3.2 addRef()

```
BRefData * BRefData::addRef ( )
```

Increment the reference counter.

7.76.3.3 deleteRef()

```
int BRefData::deleteRef ( )
```

Decrement the reference counter.

7.76.3.4 data()

```
char* BRefData::data ( )  [inline]
```

Return the raw data pointer.

7.76.3.5 len()

```
int BRefData::len ( )  [inline]
```

Return the length in bytes.

7.76.3.6 operator=()

```
BRefData & BRefData::operator= (
            const BRefData & refData )
```

7.76.3.7 setLen()

```
void BRefData::setLen (
            int len )
```

Set the length in bytes. Note should only be used if orefCount = 1.

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

- BRefData.h
- BRefData.cpp

7.77 BRtc Class Reference

Realtime clock for access to the systems real time battery backed up time hardware.

```
#include <BRtc.h>
```

Public Member Functions

- BRtc ()
- ~BRtc ()
- BError init (int rate)

  Setup interrupt rate.
- void wait (int delayUs)

  Wait specified uS.

7.77.1 Detailed Description

Realtime clock for access to the systems real time battery backed up time hardware.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 BRtc()

```
BRtc::BRtc ( )
```

7.77.2.2 ~BRtc()

```
BRtc::~BRtc ( )
```

7.77.3 Member Function Documentation

7.77.3.1 init()

```
BError BRtc::init (
            int rate )
```

Setup interrupt rate.

7.77.3.2 wait()

```
void BRtc::wait (
            int delayUs )
```

Wait specified uS.

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

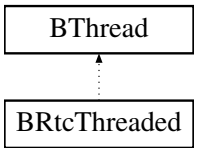
- BRtc.h
- BRtc.cpp

7.78 BRtcThreaded Class Reference

A thread safe class to access to the systems real time battery backed up time hardware.

`#include <BRtc.h>`

Inheritance diagram for BRtcThreaded:



Public Member Functions

- BRtcThreaded ()
- ~BRtcThreaded ()
- BError init (int rate)

  Setup interrupt rate.
- void wait (int delayUs)

  Wait specified uS.

7.78.1 Detailed Description

A thread safe class to access to the systems real time battery backed up time hardware.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 BRtcThreaded()

```
BRtcThreaded::BRtcThreaded ( )
```

7.78.2.2 ~BRtcThreaded()

```
BRtcThreaded::~BRtcThreaded ( )
```

7.78.3 Member Function Documentation

7.78.3.1 init()

```
BError BRtcThreaded::init (
            int rate )
```

Setup interrupt rate.

7.78.3.2 wait()

```
void BRtcThreaded::wait (
            int delayUs )
```

Wait specified uS.

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

- BRtc.h
- BRtc.cpp

7.79 BRWLock Class Reference

Thread read-write lock.

`#include <BRWLock.h>`

Public Member Functions

- BRWLock ()
- BRWLock (const BRWLock &rwlock)
- ~BRWLock ()
- int rdLock ()

  Set lock, wait if necessary.
- int tryRdLock ()

  Test the lock.
- int wrLock ()

  Set lock, wait if necessary.
- int tryWrLock ()

  Test the lock.
- int unlock ()

  Unlock the lock.
- BRWLock & operator= (const BRWLock &rwlock)

7.79.1 Detailed Description

Thread read-write lock.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 BRWLock() [1/2]

```
BRWLock::BRWLock ( )
```

7.79.2.2 BRWLock() [2/2]

```
BRWLock::BRWLock (
            const BRWLock & rwlock )
```

7.79.2.3 ~BRWLock()

```
BRWLock::~BRWLock ( )
```

7.79.3 Member Function Documentation

7.79.3.1 rdLock()

```
int BRWLock::rdLock ( )
```

Set lock, wait if necessary.

7.79.3.2 tryRdLock()

```
int BRWLock::tryRdLock ( )
```

Test the lock.

7.79.3.3 wrLock()

```
int BRWLock::wrLock ( )
```

Set lock, wait if necessary.

7.79.3.4 tryWrLock()

```
int BRWLock::tryWrLock ( )
```

Test the lock.

7.79.3.5 unlock()

```
int BRWLock::unlock ( )
```

Unlock the lock.

7.79.3.6 operator=()

```
BRWLock & BRWLock::operator= (
            const BRWLock & rwlock )
```

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

- BRWLock.h
- BRWLock.cpp

7.80 BSema Class Reference

Sempahore class.

```
#include <BSema.h>
```

Public Member Functions

- BSema (int value=0)
- BSema (const BSema &sema)
- ~BSema ()
- int post ()

  Post condition.
- int wait ()

  Wait for contition.
- int timedWait (int timeUs)

  Wait for condition with timeout.
- int tryWait ()

  Test for the condition.
- int getValue () const
- BSema & operator= (const BSema &sema)

7.80.1 Detailed Description

Sempahore class.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 BSema() [1/2]

```
BSema::BSema (
            int value = 0 )
```

7.80.2.2 BSema() [2/2]

```
BSema::BSema (
            const BSema & sema )
```

7.80.2.3 ~BSema()

```
BSema::~BSema ( )
```

7.80.3 Member Function Documentation

7.80.3.1 post()

```
int BSema::post ( )
```

Post condition.

7.80.3.2 wait()

```
int BSema::wait ( )
```

Wait for contition.

7.80.3.3 timedWait()

```
int BSema::timedWait (
            int timeUs )
```

Wait for condition with timeout.

7.80.3.4 tryWait()

```
int BSema::tryWait ( )
```

Test for the condition.

7.80.3.5 getValue()

```
int BSema::getValue ( ) const
```

7.80.3.6 operator=()

```
BSema & BSema::operator= (
            const BSema & sema )
```

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

- BSema.h
- BSema.cpp

7.81 BSemaphore Class Reference

Base Semaphore class.

`#include <BSemaphore.h>`

Public Member Functions

- BSemaphore ()
- BSemaphore (const BSemaphore &semaphore)
- ~BSemaphore ()
- Bool wait (BTimeout timeoutUs=BTimeoutForever)

  Wait for the semaphore.
- void set ()

  Set the semaphore.
- int getValue () const
- BSemaphore & operator= (const BSemaphore &semaphore)

7.81.1 Detailed Description

Base Semaphore class.

7.81.2 Constructor & Destructor Documentation

7.81.2.1 BSemaphore() [1/2]

```
BSemaphore::BSemaphore ( )
```

7.81.2.2 BSemaphore() [2/2]

```
BSemaphore::BSemaphore (
            const BSemaphore & semaphore )
```

7.81.2.3 ~BSemaphore()

```
BSemaphore::~BSemaphore ( )
```

7.81.3 Member Function Documentation

7.81.3.1 wait()

```
Bool BSemaphore::wait (
            BTimeout timeoutUs = BTimeoutForever )
```

Wait for the semaphore.

7.81.3.2 set()

```
void BSemaphore::set ( )
```

Set the semaphore.

7.81.3.3 getValue()

```
int BSemaphore::getValue ( ) const
```

7.81.3.4 operator=()

```
BSemaphore & BSemaphore::operator= (
            const BSemaphore & semaphore )
```

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

- BSemaphore.h
- BSemaphore.cpp

7.82 BSemaphoreBool Class Reference

Boolean semaphore.

```
#include <BSemaphore.h>
```

Public Member Functions

- BSemaphoreBool ()
- BSemaphoreBool (const BSemaphoreBool &semaphore)
- ~BSemaphoreBool ()
- void set (Bool on=1)
- void clear ()
- Bool wait (Bool v=1, BTimeout timeoutUs=BTimeoutForever)

  Wait for the semaphore.
- Bool value ()
- operator int ()
- int operator== (Bool on)
- BSemaphoreBool & operator= (Bool on)

7.82.1 Detailed Description

Boolean semaphore.

7.82.2 Constructor & Destructor Documentation

7.82.2.1 BSemaphoreBool() [1/2]

```
BSemaphoreBool::BSemaphoreBool ( )
```

7.82.2.2 BSemaphoreBool() [2/2]

```
BSemaphoreBool::BSemaphoreBool (
            const BSemaphoreBool & semaphore )
```

7.82.2.3 ~BSemaphoreBool()

```
BSemaphoreBool::~BSemaphoreBool ( )
```

7.82.3 Member Function Documentation

7.82.3.1 set()

```
void BSemaphoreBool::set (
            Bool on = 1 )
```

7.82.3.2 clear()

```
void BSemaphoreBool::clear ( )
```

7.82.3.3 wait()

```
Bool BSemaphoreBool::wait (
            Bool v = 1,
            BTimeout timeoutUs = BTimeoutForever )
```

Wait for the semaphore.

7.82.3.4 value()

```
Bool BSemaphoreBool::value ( )
```

7.82.3.5 operator int()

```
BSemaphoreBool::operator int ( )
```

7.82.3.6 operator==()

```
int BSemaphoreBool::operator== (
            Bool on )
```

7.82.3.7 operator=()

```
BSemaphoreBool & BSemaphoreBool::operator= (
            Bool on )
```

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

- BSemaphore.h
- BSemaphore.cpp

7.83 BSemaphoreCount Class Reference

Integer counting semaphore.

`#include <BSemaphore.h>`

Public Member Functions

- BSemaphoreCount ()
- BSemaphoreCount (const BSemaphoreCount &semaphore)
- ~BSemaphoreCount ()
- void setValue (BUInt v)
- void add (int v=1)

  Set the semaphore.
- Bool wait (BUInt v=1, BTimeout timeoutUs=BTimeoutForever)

  Wait for the semaphore.
- Bool take (BUInt v=1, BTimeout timeoutUs=BTimeoutForever)

  Take for the semaphore.
- BUInt value ()
- BSemaphoreCount & operator= (const BSemaphoreCount &semaphore)

7.83.1 Detailed Description

Integer counting semaphore.

7.83.2 Constructor & Destructor Documentation

7.83.2.1 BSemaphoreCount() [1/2]

```
BSemaphoreCount::BSemaphoreCount ( )
```

7.83.2.2 BSemaphoreCount() [2/2]

```
BSemaphoreCount::BSemaphoreCount (
            const BSemaphoreCount & semaphore )
```

7.83.2.3 ~BSemaphoreCount()

```
BSemaphoreCount::~BSemaphoreCount ( )
```

7.83.3 Member Function Documentation

7.83.3.1 setValue()

```
void BSemaphoreCount::setValue (
            BUInt v )
```

7.83.3.2 add()

```
void BSemaphoreCount::add (
            int v = 1 )
```

Set the semaphore.

7.83.3.3 wait()

```
Bool BSemaphoreCount::wait (
            BUInt v = 1,
            BTimeout timeoutUs = BTimeoutForever )
```

Wait for the semaphore.

7.83.3.4 take()

```
Bool BSemaphoreCount::take (
            BUInt v = 1,
            BTimeout timeoutUs = BTimeoutForever )
```

Take for the semaphore.

7.83.3.5 value()

```
BUInt BSemaphoreCount::value ( )
```

7.83.3.6 operator=()

```
BSemaphoreCount & BSemaphoreCount::operator= (
            const BSemaphoreCount & semaphore )
```

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

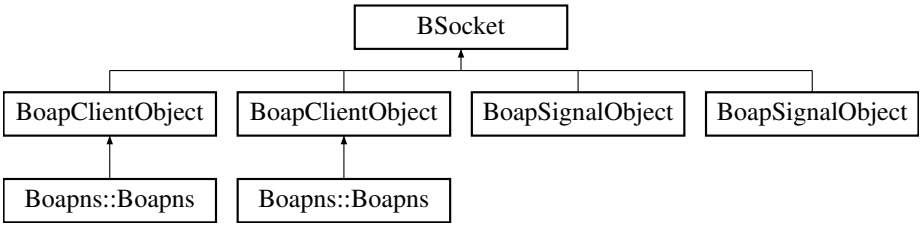
- BSemaphore.h
- BSemaphore.cpp

7.84 BSocket Class Reference

A network communications socket.

`#include <BSocket.h>`

Inheritance diagram for BSocket:



Public Types

- enum NType { STREAM , DGRAM }
- enum Priority { PriorityLow , PriorityNormal , PriorityHigh }

Public Member Functions

- BSocket ()
- BSocket (int fd)
- BSocket (NType type)
- BSocket (int domain, int type, int protocol)
- ~BSocket ()
- BError init (int domain, int type, int protocol)
- BError init (NType type)
- void setFd (int fd)
- int getFd ()
- BError bind (const BSocketAddress &add)
- BError connect (const BSocketAddress &add)
- BError shutdown (int how)
- BError close ()
- BError listen (int backlog=5)
- BError accept (int &fd)
- BError accept (int &fd, BSocketAddress &address)
- BError send (const void *buf, BSize nbytes, BSize &nbytesSent, int flags=0)
- BError sendTo (const BSocketAddress &address, const void *buf, BSize nbytes, BSize &nbytesSent, int flags=0)
- BError sendChunks (const BDataChunk *chunks, BSize nChunks, BSize &nbytesSent, int flags=0)
- BError recv (void *buf, BSize maxbytes, BSize &nbytesRecv, int flags=0)
- BError recvFrom (BSocketAddress &address, void *buf, BSize maxbytes, BSize &nbytesRecv, int flags=0)
- BError recvWithTimeout (void *buf, BSize maxbytes, BSize &nbytesRecv, int timeout, int flags=0)
- BError recvFromWithTimeout (BSocketAddress &address, void *buf, BSize maxbytes, BSize &nbytesRecv, int timeout, int flags=0)
- BUInt recvAvailable ()
- BError setSockOpt (int level, int optname, void *optval, unsigned int optlen)
- BError getSockOpt (int level, int optname, void *optval, unsigned int *optlen)
- BError setReuseAddress (int on)
- BError setBroadCast (int on)
- BError setPriority (Priority priority)
- BError getMTU (uint32_t &mtu)
- BError getAddress (BSocketAddress &address)

7.84.1 Detailed Description

A network communications socket.

7.84.2 Member Enumeration Documentation

7.84.2.1 NType

`enum BSocket::NType`

Enumerator

STREAM	
DGRAM	

7.84.2.2 Priority

```
enum BSocket::Priority
```

Enumerator

PriorityLow	
PriorityNormal	
PriorityHigh	

7.84.3 Constructor & Destructor Documentation

7.84.3.1 BSocket() [1/4]

```
BSocket::BSocket ( )
```

7.84.3.2 BSocket() [2/4]

```
BSocket::BSocket (
            int fd )
```

7.84.3.3 BSocket() [3/4]

```
BSocket::BSocket (
            NType type )
```

7.84.3.4 BSocket() [4/4]

```
BSocket::BSocket (
            int domain,
            int type,
            int protocol )
```

7.84.3.5 ~BSocket()

```
BSocket::~BSocket ( )
```

7.84.4 Member Function Documentation

7.84.4.1 init() [1/2]

```
BError BSocket::init (
            int domain,
            int type,
            int protocol )
```

7.84.4.2 init() [2/2]

```
BError BSocket::init (
            NType type )
```

7.84.4.3 setFd()

```
void BSocket::setFd (
            int fd )
```

7.84.4.4 getFd()

```
int BSocket::getFd ( )
```

7.84.4.5 bind()

```
BError BSocket::bind (
            const BSocketAddress & add )
```

7.84.4.6 connect()

```
BError BSocket::connect (
            const BSocketAddress & add )
```

7.84.4.7 shutdown()

```
BError BSocket::shutdown (
            int how )
```

7.84.4.8 close()

```
BError BSocket::close ( )
```

7.84.4.9 listen()

```
BError BSocket::listen (
            int backlog = 5 )
```

7.84.4.10 accept() [1/2]

```
BError BSocket::accept (
            int & fd )
```

7.84.4.11 accept() [2/2]

```
BError BSocket::accept (
            int & fd,
            BSocketAddress & address )
```

7.84.4.12 send()

```
BError BSocket::send (
            const void * buf,
            BSize nbytes,
            BSize & nbytesSent,
            int flags = 0 )
```

7.84.4.13 sendTo()

```
BError BSocket::sendTo (
            const BSocketAddress & address,
            const void * buf,
            BSize nbytes,
            BSize & nbytesSent,
            int flags = 0 )
```

7.84.4.14 sendChunks()

```
BError BSocket::sendChunks (
            const BDataChunk * chunks,
            BSize nChunks,
            BSize & nbytesSent,
            int flags = 0 )
```

7.84.4.15 recv()

```
BError BSocket::recv (
            void * buf,
            BSize maxbytes,
            BSize & nbytesRecv,
            int flags = 0 )
```

7.84.4.16 recvFrom()

```
BError BSocket::recvFrom (
            BSocketAddress & address,
            void * buf,
            BSize maxbytes,
            BSize & nbytesRecv,
            int flags = 0 )
```

7.84.4.17 recvWithTimeout()

```
BError BSocket::recvWithTimeout (
            void * buf,
            BSize maxbytes,
            BSize & nbytesRecv,
            int timeout,
            int flags = 0 )
```

7.84.4.18 recvFromWithTimeout()

```
BError BSocket::recvFromWithTimeout (
            BSocketAddress & address,
            void * buf,
            BSize maxbytes,
            BSize & nbytesRecv,
            int timeout,
            int flags = 0 )
```

7.84.4.19 recvAvailable()

```
BUInt BSocket::recvAvailable ( )
```

7.84.4.20 setSockOpt()

```
BError BSocket::setSockOpt (
            int level,
            int optname,
            void * optval,
            unsigned int optlen )
```

7.84.4.21 getSockOpt()

```
BError BSocket::getSockOpt (
            int level,
            int optname,
            void * optval,
            unsigned int * optlen )
```

7.84.4.22 setReuseAddress()

```
BError BSocket::setReuseAddress (
            int on )
```

7.84.4.23 setBroadCast()

```
BError BSocket::setBroadCast (
            int on )
```

7.84.4.24 setPriority()

```
BError BSocket::setPriority (
            Priority priority )
```

7.84.4.25 getMTU()

```
BError BSocket::getMTU (
            uint32_t & mtu )
```

7.84.4.26 getAddress()

```
BError BSocket::getAddress (
            BSocketAddress & address )
```

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

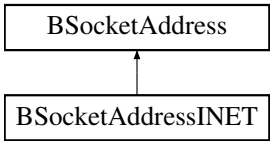
- BSocket.h
- BSocket.cpp

7.85 BSocketAddress Class Reference

Socket Address.

`#include <BSocket.h>`

Inheritance diagram for BSocketAddress:



Public Types

- typedef struct sockaddr SockAddr

Public Member Functions

- BSocketAddress ()
- BSocketAddress (const BSocketAddress &add)
- BSocketAddress (SockAddr *address, int len)
- ~BSocketAddress ()
- BError set (SockAddr *address, int len)
- const SockAddr * raw () const
- int len () const
- BString getString () const

  Return string version of address <ip>:<port>
- BSocketAddress & operator= (const BSocketAddress &add)
- operator const SockAddr * () const
- int operator== (const BSocketAddress &add) const
- int operator!= (const BSocketAddress &add) const

7.85.1 Detailed Description

Socket Address.

7.85.2 Member Typedef Documentation

7.85.2.1 SockAddr

```
typedef struct sockaddr BSocketAddress::SockAddr
```

7.85.3 Constructor & Destructor Documentation

7.85.3.1 BSocketAddress() [1/3]

```
BSocketAddress::BSocketAddress ( )
```

7.85.3.2 BSocketAddress() [2/3]

```
BSocketAddress::BSocketAddress (
            const BSocketAddress & add )
```

7.85.3.3 BSocketAddress() [3/3]

```
BSocketAddress::BSocketAddress (
            SockAddr * address,
            int len )
```

7.85.3.4 ~BSocketAddress()

```
BSocketAddress::~BSocketAddress ( )
```

7.85.4 Member Function Documentation

7.85.4.1 set()

```
BError BSocketAddress::set (
            SockAddr * address,
            int len )
```

7.85.4.2 raw()

```
const BSocketAddress::SockAddr * BSocketAddress::raw ( ) const
```

7.85.4.3 len()

```
int BSocketAddress::len ( ) const
```

7.85.4.4 getString()

```
BString BSocketAddress::getString ( ) const
```

Return string version of address <ip>:<port>

7.85.4.5 operator=()

```
BSocketAddress & BSocketAddress::operator= (
            const BSocketAddress & add )
```

7.85.4.6 operator const SockAddr *()

```
BSocketAddress::operator const SockAddr * ( ) const  [inline]
```

7.85.4.7 operator==()

```
int BSocketAddress::operator== (
            const BSocketAddress & add ) const
```

7.85.4.8 operator"!=()

```
int BSocketAddress::operator!= (
            const BSocketAddress & add ) const
```

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

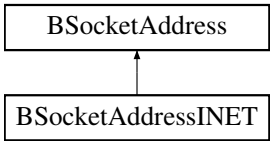
- BSocket.h
- BSocket.cpp

7.86 BSocketAddressINET Class Reference

IPV4 aware socket address.

```
#include <BSocket.h>
```

Inheritance diagram for BSocketAddressINET:



Public Types

- typedef struct sockaddr_in SockAddrIP

Public Member Functions

- BError set (BString hostName, uint32_t port)
- BError set (uint32_t address, uint32_t port)
- BError set (BString hostName, BString service, BString type)
- void setPort (uint32_t port)
- uint32_t address ()

  Returns socket ip address.
- uint32_t port ()

  Returns socket port.
- BString getString ()

  Return string version of address <ip>:<port>

Static Public Member Functions

- static BString getHostName ()

  Get this hosts network name.
- static BList< uint32_t > getIpAddresses ()

  Get a list of all the IP addresses of this host.
- static BList< BString > getIpAddressList ()

  Get a list of all the IP addresses of this host under hostname.
- static BList< BString > getIpAddressListAll ()

  Get a list of all the IP addresses of this host looking at physical interfaces.

7.86.1 Detailed Description

IPV4 aware socket address.

7.86.2 Member Typedef Documentation

7.86.2.1 SockAddrIP

```
typedef struct sockaddr_in BSocketAddressINET::SockAddrIP
```

7.86.3 Member Function Documentation

7.86.3.1 set() [1/3]

```
BError BSocketAddressINET::set (
            BString hostName,
            uint32_t port )
```

7.86.3.2 set() [2/3]

```
BError BSocketAddressINET::set (
            uint32_t address,
            uint32_t port )
```

7.86.3.3 set() [3/3]

```
BError BSocketAddressINET::set (
            BString hostName,
            BString service,
            BString type )
```

7.86.3.4 setPort()

```
void BSocketAddressINET::setPort (
            uint32_t port )
```

7.86.3.5 address()

```
uint32_t BSocketAddressINET::address ( )
```

Returns socket ip address.

7.86.3.6 port()

```
uint32_t BSocketAddressINET::port ( )
```

Returns socket port.

7.86.3.7 getString()

```
BString BSocketAddressINET::getString ( )
```

Return string version of address <ip>:<port>

7.86.3.8 getHostName()

```
BString BSocketAddressINET::getHostName ( )  [static]
```

Get this hosts network name.

7.86.3.9 getIpAddresses()

```
BList< uint32_t > BSocketAddressINET::getIpAddresses ( )  [static]
```

Get a list of all the IP addresses of this host.

7.86.3.10 getIpAddressList()

```
BList< BString > BSocketAddressINET::getIpAddressList ( )  [static]
```

Get a list of all the IP addresses of this host under hostname.

7.86.3.11 getIpAddressListAll()

```
BList< BString > BSocketAddressINET::getIpAddressListAll ( )  [static]
```

Get a list of all the IP addresses of this host looking at physical interfaces.

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

- BSocket.h
- BSocket.cpp

7.87 BSpi Class Reference

BSpi class for accessing SPI hardware devices.

```
#include <BSpi.h>
```

Public Types

- enum Mode { Mode0 = 0 , Mode1 = 1 , Mode2 = 2 , Mode3 = 3 }

Public Member Functions

- BSpi ()
- BError init (BString devName, BUInt speed=1000000, Mode mode=Mode1, Bool csActive=0)
- BError transact (BUInt8 dev, void *txBuf, int txLen, int pad, void *rxBuf, int rxLen)

7.87.1 Detailed Description

BSpi class for accessing SPI hardware devices.

7.87.2 Member Enumeration Documentation

7.87.2.1 Mode

```
enum BSpi::Mode
```

Enumerator

Mode0	
Mode1	
Mode2	
Mode3	

7.87.3 Constructor & Destructor Documentation

7.87.3.1 BSpi()

```
BSpi::BSpi ( )
```

7.87.4 Member Function Documentation

7.87.4.1 init()

```
BError BSpi::init (
            BString devName,
            BUInt speed = 1000000,
            Mode mode = Mode1,
            Bool csActive = 0 )
```

7.87.4.2 transact()

```
BError BSpi::transact (
            BUInt8 dev,
            void * txBuf,
            int txLen,
            int pad,
            void * rxBuf,
            int rxLen )
```

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

- BSpi.h
- BSpi.cpp

7.88 BString Class Reference

This class stores and manipulates ASCII strings.

```
#include <BString.h>
```

Public Member Functions

- BString ()
- BString (const BString &string)
- BString (const char *str)
- BString (const char *str, unsigned int len)
- BString (char ch)
- BString (BInt v)
- BString (BUInt v)
- BString (BUInt64 v)
- BString (double v)
- ~BString ()
- BString copy () const

  Return an independant copy.
- int len () const

  Length of string.
- const char * retStr () const

  Ptr to char representation.*
- const char * str () const

Ptr to char representation.*

- char * retStrDup () const

  Ptr to newly malloc'd char.*
- int retInt () const

  Return string as a int.
- unsigned int retUInt () const

  Return string as a int.
- double retDouble () const

  Return string as a double.
- BFloat64 retFloat64 () const

  Return string as a BFloat64.
- int compare (const BString &string) const

  Compare strings. Returns 0 for match.
- int compareWild (const BString &string) const

  Compare string to string with wildcards . Returns 0 for match.
- int compareWildExpression (const BString &string) const

  Compare string to space deliminated patterns. Returns 0 for match.
- int compareRegex (const BString &pattern, int ignoreCase=0) const

  Compare strings. Returns 1 for match.
- BString & truncate (int len)

  Truncate to length len.
- BString & pad (int len)

  Pad to length len.
- void clear ()

  Clear the string.
- BString & toUpper ()

  Convert to uppercase.
- BString & toLower ()

  Convert to lowercase.
- BString lowerFirst ()

  Return string with lowercase first character.
- void removeNL ()

  Remove if present NL from last char.
- BString justify (int leftMargin, int width)

  Justify the string to the given width.
- BString fixedLen (int length, int rightJustify=0)

  return string formated to fixed length
- BString firstLine ()

  Return first line.
- BString translateChar (char ch, BString replace=" ")

  Translate character converting them to the given string.
- BString reverse () const

  Reverse character order.
- BString subString (int start, int len) const

  Returns substring.
- int del (int start, int len)

  Delete substring.
- int insert (int start, BString str)

  Insert substring.
- int append (const BString &str)

  Append a string.

- BString add (const BString &str) const

  Add strings returning result.
- BString & printf (const char *fmt,...)

  Formated print into the string.
- int find (char ch) const

  Find ch in string searching forwards.
- int find (BString str) const

  Find string in string searching forwards.
- int findReverse (char ch) const

  Find ch in string searching backwards.
- BString csvEncode () const

  Encode a string for CSV.
- BString & csvDecode (const BString str)

  Decode a string from CSV.
- BString base64Encode () const

  Encode a string to base64.
- BError base64Decode (BString &str) const

  Decode a string from base64.
- BList< BString > getTokenList (BString separators)

  Break string into tokens.
- BList< BString > getTokenList (char separator)

  Break string into tokens.
- BString removeSeparators (BString separators)

  Remove any char from sepatators from string.
- BString pullToken (BString terminators)

  Pull token from start of string.
- BString pullSeparators (BString separators)

  Pull separators from start of string.
- BString pullWord ()

  Pull a word out of the head of the string.
- BString pullLine ()

  Pull a line out of the head of the string.
- BList< BString > split (char splitChar)

  Split string into an array based on the character separator.
- BString dirname ()

  Return the directory component if the string is a file path name.
- BString basename ()

  Return the file name component if the string is a file path name.
- BString extension ()

  Return the file extension component if the string is a file path name.
- BUInt32 hash () const

  Create a 32bit hash number for the string.
- char & get (int pos)

  Return the character at the position give. Will print an error message and raise the SIGABORT exception if out of range.
- const char & get (int pos) const

  Return the character at the position give. Will print an error message and raise the SIGABORT exception if out of range.
- BString & operator= (const BString &string)
- char & operator[ ] (int pos)
- int operator== (const BString &s) const

- int operator== (const char *s) const
- int operator> (const BString &s) const
- int operator> (const char *s) const
- int operator< (const BString &s) const
- int operator< (const char *s) const
- int operator>= (const BString &s) const
- int operator<= (const BString &s) const
- int operator!= (const BString &s) const
- int operator!= (const char *s) const
- BString operator+ (const BString &s) const
- BString operator+ (const char *s) const
- BString operator+= (const BString &s)
- BString operator+= (const char *s)
- BString operator+ (char ch) const
- BString operator+ (BInt i) const
- BString operator+ (BUInt i) const
- BString operator+ (BUInt64 i) const
- operator const char * () const
- BString field (int field) const

  Depreciated function.
- char ** fields ()

  Depreciated function.

Static Public Member Functions

- static BString convert (char ch)

  Converts char to string.
- static BString convert (BInt value)

  Converts int to string.
- static BString convert (BUInt value)

  Converts uint to string.
- static BString convert (double value, int eFormat=0)

  Converts double to string.
- static BString convert (BUInt64 value)

  Converts long long to string.
- static BString convertHex (BInt value)

  Converts int to string as hex value.
- static BString convertHex (BUInt value)

  Converts uint to string as hex value.

Protected Attributes

- BRefData * ostr

  This is the reference counted string storage.

7.88.1 Detailed Description

This class stores and manipulates ASCII strings.

The BString class is designed for the storage and manipulation of variable length ASCII strings. It uses the BRefData class to store the null terminated strings in shared and references counted shared memory areas on the heap. This efficient method reduces memory allocation, deallocation and copying. Copying or just passing a string in and out of a function becomes a simple pointer copy with the string data reference count incremented.

You can convert a BStirng to a const char* using the BString::str() method. You can create a BString from a char* with its constructor BString(const char*) which is used implicitly for conversions. The BString also supports conversion to and from Qt QStrings if the Qt headers are included before the BString.h header. The BString also supports input and output from std::iostream objects.

As well as its constructor function, BString supports a number of explicity convert() functions that can convert basic types such as integers to a BString. There are also some ret*() functions to parse the BString for integer and floating point values.

The operators: =, +, <, >, >=, <=, ==, != operate as you would expect with the +" appending strings.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 BString() [1/9]

```
BString::BString ( )
```

7.88.2.2 BString() [2/9]

```
BString::BString (
            const BString & string )
```

7.88.2.3 BString() [3/9]

```
BString::BString (
            const char * str )
```

7.88.2.4 BString() [4/9]

```
BString::BString (
            const char * str,
            unsigned int len )
```

7.88.2.5 BString() [5/9]

```
BString::BString (
            char ch )
```

7.88.2.6 BString() [6/9]

```
BString::BString (
            BInt v )
```

7.88.2.7 BString() [7/9]

```
BString::BString (
            BUInt v )
```

7.88.2.8 BString() [8/9]

```
BString::BString (
            BUInt64 v )
```

7.88.2.9 BString() [9/9]

```
BString::BString (
            double v )
```

7.88.2.10 ~BString()

```
BString::~BString ( )
```

7.88.3 Member Function Documentation

7.88.3.1 convert() [1/5]

```
BString BString::convert (
            char ch )  [static]
```

Converts char to string.

7.88.3.2 convert() [2/5]

```
BString BString::convert (
            BInt value )  [static]
```

Converts int to string.

7.88.3.3 convert() [3/5]

```
BString BString::convert (
            BUInt value )  [static]
```

Converts uint to string.

7.88.3.4 convert() [4/5]

```
BString BString::convert (
            double value,
            int eFormat = 0 )  [static]
```

Converts double to string.

7.88.3.5 convert() [5/5]

```
BString BString::convert (
            BUInt64 value )  [static]
```

Converts long long to string.

7.88.3.6 convertHex() [1/2]

```
BString BString::convertHex (
            BInt value )  [static]
```

Converts int to string as hex value.

7.88.3.7 convertHex() [2/2]

```
BString BString::convertHex (
            BUInt value )  [static]
```

Converts uint to string as hex value.

7.88.3.8 copy()

```
BString BString::copy ( ) const
```

Return an independant copy.

7.88.3.9 len()

```
int BString::len ( ) const
```

Length of string.

7.88.3.10 retStr()

```
const char * BString::retStr ( ) const
```

Ptr to char* representation.

7.88.3.11 str()

```
const char * BString::str ( ) const
```

Ptr to char* representation.

7.88.3.12 retStrDup()

```
char * BString::retStrDup ( ) const
```

Ptr to newly malloc'd char*.

7.88.3.13 retInt()

```
int BString::retInt ( ) const
```

Return string as a int.

7.88.3.14 retUInt()

```
unsigned int BString::retUInt ( ) const
```

Return string as a int.

7.88.3.15 retDouble()

```
double BString::retDouble ( ) const
```

Return string as a double.

7.88.3.16 retFloat64()

```
BFloat64 BString::retFloat64 ( ) const
```

Return string as a BFloat64.

7.88.3.17 compare()

```
int BString::compare (
            const BString & string ) const
```

Compare strings. Returns 0 for match.

7.88.3.18 compareWild()

```
int BString::compareWild (
            const BString & string ) const
```

Compare string to string with wildcards . Returns 0 for match.

7.88.3.19 compareWildExpression()

```
int BString::compareWildExpression (
            const BString & string ) const
```

Compare string to space deliminated patterns. Returns 0 for match.

7.88.3.20 compareRegex()

```
int BString::compareRegex (
            const BString & pattern,
            int ignoreCase = 0 ) const
```

Compare strings. Returns 1 for match.

7.88.3.21 truncate()

```
BString & BString::truncate (
            int len )
```

Truncate to length len.

7.88.3.22 pad()

```
BString & BString::pad (
            int len )
```

Pad to length len.

7.88.3.23 clear()

```
void BString::clear ( )
```

Clear the string.

7.88.3.24 toUpper()

```
BString & BString::toUpper ( )
```

Convert to uppercase.

7.88.3.25 toLower()

```
BString & BString::toLower ( )
```

Convert to lowercase.

7.88.3.26 lowerFirst()

```
BString BString::lowerFirst ( )
```

Return string with lowercase first character.

7.88.3.27 removeNL()

```
void BString::removeNL ( )
```

Remove if present NL from last char.

7.88.3.28 justify()

```
BString BString::justify (
            int leftMargin,
            int width )
```

Justify the string to the given width.

7.88.3.29 fixedLen()

```
BString BString::fixedLen (
            int length,
            int rightJustify = 0 )
```

return string formated to fixed length

7.88.3.30 firstLine()

```
BString BString::firstLine ( )
```

Return first line.

7.88.3.31 translateChar()

```
BString BString::translateChar (
            char ch,
            BString replace = " " )
```

Translate character converting them to the given string.

7.88.3.32 reverse()

```
BString BString::reverse ( ) const
```

Reverse character order.

7.88.3.33 subString()

```
BString BString::subString (
            int start,
            int len ) const
```

Returns substring.

7.88.3.34 del()

```
int BString::del (
            int start,
            int len )
```

Delete substring.

7.88.3.35 insert()

```
int BString::insert (
            int start,
            BString str )
```

Insert substring.

7.88.3.36 append()

```
int BString::append (
            const BString & str )
```

Append a string.

7.88.3.37 add()

```
BString BString::add (
            const BString & str ) const
```

Add strings returning result.

7.88.3.38 printf()

```
BString & BString::printf (
            const char * fmt,
             ...  )
```

Formated print into the string.

7.88.3.39 find() [1/2]

```
int BString::find (
            char ch ) const
```

Find ch in string searching forwards.

7.88.3.40 find() [2/2]

```
int BString::find (
            BString str ) const
```

Find string in string searching forwards.

7.88.3.41 findReverse()

```
int BString::findReverse (
            char ch ) const
```

Find ch in string searching backwards.

7.88.3.42 csvEncode()

```
BString BString::csvEncode ( ) const
```

Encode a string for CSV.

7.88.3.43 csvDecode()

```
BString & BString::csvDecode (
            const BString str )
```

Decode a string from CSV.

7.88.3.44 base64Encode()

```
BString BString::base64Encode ( ) const
```

Encode a string to base64.

7.88.3.45 base64Decode()

```
BError BString::base64Decode (
            BString & str ) const
```

Decode a string from base64.

7.88.3.46 getTokenList() [1/2]

```
BList< BString > BString::getTokenList (
            BString separators )
```

Break string into tokens.

7.88.3.47 getTokenList() [2/2]

```
BList< BString > BString::getTokenList (
            char separator )
```

Break string into tokens.

7.88.3.48 removeSeparators()

```
BString BString::removeSeparators (
            BString separators )
```

Remove any char from sepatators from string.

7.88.3.49 pullToken()

```
BString BString::pullToken (
            BString terminators )
```

Pull token from start of string.

7.88.3.50 pullSeparators()

```
BString BString::pullSeparators (
            BString separators )
```

Pull separators from start of string.

7.88.3.51 pullWord()

```
BString BString::pullWord ( )
```

Pull a word out of the head of the string.

7.88.3.52 pullLine()

```
BString BString::pullLine ( )
```

Pull a line out of the head of the string.

7.88.3.53 split()

```
BList< BString > BString::split (
            char splitChar )
```

Split string into an array based on the character separator.

7.88.3.54 dirname()

```
BString BString::dirname ( )
```

Return the directory component if the string is a file path name.

7.88.3.55 basename()

```
BString BString::basename ( )
```

Return the file name component if the string is a file path name.

7.88.3.56 extension()

```
BString BString::extension ( )
```

Return the file extension component if the string is a file path name.

7.88.3.57 hash()

```
BUInt32 BString::hash ( ) const
```

Create a 32bit hash number for the string.

7.88.3.58 get() [1/2]

```
char & BString::get (
            int pos )
```

Return the character at the position give. Will print an error message and raise the SIGABORT exception if out of range.

7.88.3.59 get() [2/2]

```
const char & BString::get (
            int pos ) const
```

Return the character at the position give. Will print an error message and raise the SIGABORT exception if out of range.

7.88.3.60 operator=()

```
BString & BString::operator= (
            const BString & string )
```

7.88.3.61 operator[]()

```
char & BString::operator[] (
            int pos )
```

7.88.3.62 operator==() [1/2]

```
int BString::operator== (
            const BString & s ) const  [inline]
```

7.88.3.63 operator==() [2/2]

```
int BString::operator== (
            const char * s ) const  [inline]
```

7.88.3.64 operator>() [1/2]

```
int BString::operator> (
            const BString & s ) const  [inline]
```

7.88.3.65 operator>() [2/2]

```
int BString::operator> (
            const char * s ) const  [inline]
```

7.88.3.66 operator<() [1/2]

```
int BString::operator< (
            const BString & s ) const  [inline]
```

7.88.3.67 operator<() [2/2]

```
int BString::operator< (
            const char * s ) const  [inline]
```

7.88.3.68 operator>=()

```
int BString::operator>= (
            const BString & s ) const  [inline]
```

7.88.3.69 operator<=()

```
int BString::operator<= (
            const BString & s ) const  [inline]
```

7.88.3.70 operator"!=() [1/2]

```
int BString::operator!= (
            const BString & s ) const  [inline]
```

7.88.3.71 operator"!=() [2/2]

```
int BString::operator!= (
            const char * s ) const  [inline]
```

7.88.3.72 operator+() [1/6]

```
BString BString::operator+ (
            const BString & s ) const  [inline]
```

7.88.3.73 operator+() [2/6]

```
BString BString::operator+ (
            const char * s ) const  [inline]
```

7.88.3.74 operator+=() [1/2]

```
BString BString::operator+= (
            const BString & s )  [inline]
```

7.88.3.75 operator+=() [2/2]

```
BString BString::operator+= (
            const char * s )  [inline]
```

7.88.3.76 operator+() [3/6]

```
BString BString::operator+ (
            char ch ) const  [inline]
```

7.88.3.77 operator+() [4/6]

```
BString BString::operator+ (
            BInt i ) const  [inline]
```

7.88.3.78 operator+() [5/6]

```
BString BString::operator+ (
            BUInt i ) const  [inline]
```

7.88.3.79 operator+() [6/6]

```
BString BString::operator+ (
            BUInt64 i ) const  [inline]
```

7.88.3.80 operator const char *()

```
BString::operator const char * ( ) const  [inline]
```

7.88.3.81 field()

```
BString BString::field (
            int field ) const
```

Depreciated function.

7.88.3.82 fields()

```
char ** BString::fields ( )
```

Depreciated function.

7.88.4 Member Data Documentation

7.88.4.1 ostr

```
BRefData* BString::ostr  [protected]
```

This is the reference counted string storage.

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

- BString.h
- BString.cpp

7.89 BStringLocked Class Reference

Provides a basic thread locked string.

```
#include <BStringLocked.h>
```

Public Member Functions

- BStringLocked ()
- BStringLocked (const BStringLocked &s)
- BStringLocked (const BString &s)
- int len () const

  Length of string.
- operator BString () const
- BStringLocked operator+ (const BStringLocked &s) const
- BStringLocked & operator= (const BStringLocked &s)

7.89.1 Detailed Description

Provides a basic thread locked string.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 BStringLocked() [1/3]

```
BStringLocked::BStringLocked ( )  [inline]
```

7.89.2.2 BStringLocked() [2/3]

```
BStringLocked::BStringLocked (
            const BStringLocked & s )  [inline]
```

7.89.2.3 BStringLocked() [3/3]

```
BStringLocked::BStringLocked (
            const BString & s )  [inline]
```

7.89.3 Member Function Documentation

7.89.3.1 len()

```
int BStringLocked::len ( ) const  [inline]
```

Length of string.

7.89.3.2 operator BString()

```
BStringLocked::operator BString ( ) const  [inline]
```

7.89.3.3 operator+()

```
BStringLocked BStringLocked::operator+ (
            const BStringLocked & s ) const  [inline]
```

7.89.3.4 operator=()

```
BStringLocked& BStringLocked::operator= (
            const BStringLocked & s )  [inline]
```

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

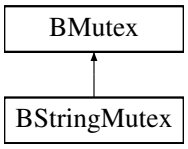
- BStringLocked.h

7.90 BStringMutex Class Reference

Thread locked string internal mutex.

`#include <BStringLocked.h>`

Inheritance diagram for BStringMutex:



Public Member Functions

- BStringMutex ()

Additional Inherited Members

7.90.1 Detailed Description

Thread locked string internal mutex.

7.90.2 Constructor & Destructor Documentation

7.90.2.1 BStringMutex()

`BStringMutex::BStringMutex ( ) [inline]`

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

- BStringLocked.h

7.91 BTable Class Reference

A simple string based table structure.

`#include <BTable.h>`

Public Member Functions

- BTable ()
- ~BTable ()
- void clear ()
- void setTitle (BArray< BString > title)
- void addRow (BArray< BString > data)
- BString getString ()
- void print (FILE *file=stdout)

7.91.1 Detailed Description

A simple string based table structure.

7.91.2 Constructor & Destructor Documentation

7.91.2.1 BTable()

```
BTable::BTable ( )
```

7.91.2.2 ~BTable()

```
BTable::~BTable ( )
```

7.91.3 Member Function Documentation

7.91.3.1 clear()

```
void BTable::clear ( )
```

7.91.3.2 setTitle()

```
void BTable::setTitle (
            BArray< BString > title )
```

7.91.3.3 addRow()

```
void BTable::addRow (
            BArray< BString > data )
```

7.91.3.4 getString()

```
BString BTable::getString ( )
```

7.91.3.5 print()

```
void BTable::print (
            FILE * file = stdout )
```

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

- BTable.h
- BTable.cpp

7.92 BTask Class Reference

Implements a thread of execution.

```
#include <BTask.h>
```

Public Member Functions

- BTask (const char *name="", BUInt stackSize=0, BUInt priority=1)
- virtual ~BTask ()
- void init (const char *name, BUInt stackSize=0, BUInt priority=1)
- BError start ()

  Starts the task running.
- void stop ()
- void waitForCompletion ()
- int setPriority (BUInt priority)

  Set the priority of the task: 0 upwards.
- virtual void run ()

Static Protected Member Functions

- static void * taskFunc (void *)

Protected Attributes

- const char * oname
- BUInt ostackSize
- BUInt opolicy
- BUInt opriority
- pthread_t othread
- Bool orunning

7.92.1 Detailed Description

Implements a thread of execution.

7.92.2 Constructor & Destructor Documentation

7.92.2.1 BTask()

```
BTask::BTask (
            const char * name = "",
            BUInt stackSize = 0,
            BUInt priority = 1 )
```

7.92.2.2 ~BTask()

```
BTask::~BTask ( )  [virtual]
```

7.92.3 Member Function Documentation

7.92.3.1 init()

```
void BTask::init (
            const char * name,
            BUInt stackSize = 0,
            BUInt priority = 1 )
```

7.92.3.2 start()

```
BError BTask::start ( )
```

Starts the task running.

7.92.3.3 stop()

```
void BTask::stop ( )
```

7.92.3.4 waitForCompletion()

```
void BTask::waitForCompletion ( )
```

7.92.3.5 setPriority()

```
int BTask::setPriority (
            BUInt priority )
```

Set the priority of the task: 0 upwards.

7.92.3.6 run()

```
void BTask::run ( )  [virtual]
```

7.92.3.7 taskFunc()

```
void * BTask::taskFunc (
            void * arg )  [static], [protected]
```

7.92.4 Member Data Documentation

7.92.4.1 oname

```
const char* BTask::oname  [protected]
```

7.92.4.2 ostackSize

```
BUInt BTask::ostackSize  [protected]
```

7.92.4.3 opolicy

```
BUInt BTask::opolicy  [protected]
```

7.92.4.4 opriority

```
BUInt BTask::opriority  [protected]
```

7.92.4.5 othread

```
pthread_t BTask::othread  [protected]
```

7.92.4.6 orunning

```
Bool BTask::orunning  [protected]
```

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

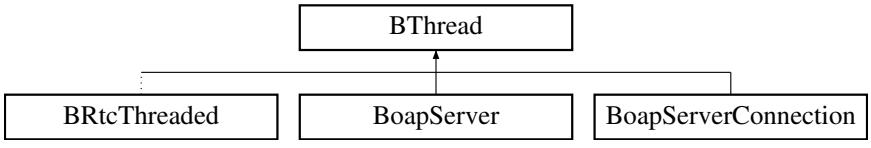
- BTask.h
- BTask.cpp

7.93 BThread Class Reference

Implements a program execution thread.

`#include <BThread.h>`

Inheritance diagram for BThread:



Public Member Functions

- BThread ()
- virtual ~BThread ()
- int setInitPriority (int policy, int priority)
- int setInitStackSize (size_t stackSize)
- int start ()
- void * result ()
- int running ()
- int setPriority (int policy, int priority)
- int cancel ()
- void * waitForCompletion ()
- pthread_t getThread ()
- virtual void * function ()

7.93.1 Detailed Description

Implements a program execution thread.

7.93.2 Constructor & Destructor Documentation

7.93.2.1 BThread()

```
BThread::BThread ( )
```

7.93.2.2 ~BThread()

```
BThread::~BThread ( )  [virtual]
```

7.93.3 Member Function Documentation

7.93.3.1 setInitPriority()

```
int BThread::setInitPriority (
            int policy,
            int priority )
```

7.93.3.2 setInitStackSize()

```
int BThread::setInitStackSize (
            size_t stackSize )
```

7.93.3.3 start()

```
int BThread::start ( )
```

7.93.3.4 result()

```
void * BThread::result ( )
```

7.93.3.5 running()

```
int BThread::running ( )
```

7.93.3.6 setPriority()

```
int BThread::setPriority (
            int policy,
            int priority )
```

7.93.3.7 cancel()

```
int BThread::cancel ( )
```

7.93.3.8 waitForCompletion()

```
void * BThread::waitForCompletion ( )
```

7.93.3.9 getThread()

```
pthread_t BThread::getThread ( )
```

7.93.3.10 function()

```
void * BThread::function ( )  [virtual]
```

Reimplemented in BoapServer.

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

- BThread.h
- BThread.cpp

7.94 BTime Class Reference

Implements a simple date/time class. Stores the date/time as a number of seconds since Unix epoch 1970-01-02T00:00:00.

```
#include <BTime.h>
```

Public Member Functions

- BTime (BUInt32 t=0)
- void set (BUInt32 seconds)

  Set the date and time.
- void set (BUInt year, BUInt month, BUInt day, BUInt hour=0, BUInt minute=0, BUInt second=0)

  Set the date and time.
- void setYearDay (BUInt year, BUInt yearDay, BUInt hour=0, BUInt minute=0, BUInt second=0)

  Set the date and time.
- void getDate (BUInt &year, BUInt &month, BUInt &day) const

  Retun the date information.
- void getTime (BUInt &hour, BUInt &minute, BUInt &second) const

  Return the time information.
- BUInt32 getSeconds () const

  Return the number of seconds.
- int isSet () const

  Check if set.
- int isLeapYear ()

  Returns if a leap year.
- void addSeconds (int seconds)

  Add the given number of seconds.
- BString getString (BString format="iso") const

  Gets the date/time in string format.
- BError setString (const BString dateTime)

Sets the date/time from string format.

- BTime utcToLocal () const

  Converts a UTC time to a local time.
- BTime localToUtc () const

  Converts a local time to UTC time.
- BString getStringLocal (BString format="iso") const

  Gets the date/time in string format.
- BError setStringLocal (const BString dateTime)

  Sets the date/time from string format.
- int operator== (const BTime &time) const
- int operator!= (const BTime &time) const
- int operator> (const BTime &time) const
- int operator>= (const BTime &time) const
- int operator< (const BTime &time) const
- int operator<= (const BTime &time) const
- BTime operator+ (int seconds) const
- BTime & operator+= (int seconds)

7.94.1 Detailed Description

Implements a simple date/time class. Stores the date/time as a number of seconds since Unix epoch 1970-01-02T00:00:00.

BTime has a range until 2106-02-07. This sets and returns datetime strings in UTC time by default. There are also ∗Local() functions to return and set using local time strings.

Uses some sepcial values. 0 - DateTime not set.

7.94.2 Constructor & Destructor Documentation

7.94.2.1 BTime()

```
BTime::BTime (
            BUInt32 t = 0 )
```

7.94.3 Member Function Documentation

7.94.3.1 set() [1/2]

```
void BTime::set (
            BUInt32 seconds )
```

Set the date and time.

7.94.3.2 set() [2/2]

```
void BTime::set (
            BUInt year,
            BUInt month,
            BUInt day,
            BUInt hour = 0,
            BUInt minute = 0,
            BUInt second = 0 )
```

Set the date and time.

7.94.3.3 setYearDay()

```
void BTime::setYearDay (
            BUInt year,
            BUInt yearDay,
            BUInt hour = 0,
            BUInt minute = 0,
            BUInt second = 0 )
```

Set the date and time.

7.94.3.4 getDate()

```
void BTime::getDate (
            BUInt & year,
            BUInt & month,
            BUInt & day ) const
```

Retun the date information.

7.94.3.5 getTime()

```
void BTime::getTime (
            BUInt & hour,
            BUInt & minute,
            BUInt & second ) const
```

Return the time information.

7.94.3.6 getSeconds()

```
BUInt32 BTime::getSeconds ( ) const
```

Return the number of seconds.

7.94.3.7 isSet()

```
int BTime::isSet ( ) const  [inline]
```

Check if set.

7.94.3.8 isLeapYear()

```
int BTime::isLeapYear ( )
```

Returns if a leap year.

7.94.3.9 addSeconds()

```
void BTime::addSeconds (
            int seconds )
```

Add the given number of seconds.

7.94.3.10 getString()

```
BString BTime::getString (
            BString format = "iso" ) const
```

Gets the date/time in string format.

7.94.3.11 setString()

```
BError BTime::setString (
            const BString dateTime )
```

Sets the date/time from string format.

7.94.3.12 utcToLocal()

```
BTime BTime::utcToLocal ( ) const
```

Converts a UTC time to a local time.

7.94.3.13 localToUtc()

```
BTime BTime::localToUtc ( ) const
```

Converts a local time to UTC time.

7.94.3.14 getStringLocal()

```
BString BTime::getStringLocal (
            BString format = "iso" ) const
```

Gets the date/time in string format.

7.94.3.15 setStringLocal()

```
BError BTime::setStringLocal (
            const BString dateTime )
```

Sets the date/time from string format.

7.94.3.16 operator==()

```
int BTime::operator== (
            const BTime & time ) const  [inline]
```

7.94.3.17 operator"!=()

```
int BTime::operator!= (
            const BTime & time ) const  [inline]
```

7.94.3.18 operator>()

```
int BTime::operator> (
            const BTime & time ) const  [inline]
```

7.94.3.19 operator>=()

```
int BTime::operator>= (
            const BTime & time ) const  [inline]
```

7.94.3.20 operator<()

```
int BTime::operator< (
            const BTime & time ) const  [inline]
```

7.94.3.21 operator<=()

```
int BTime::operator<= (
            const BTime & time ) const  [inline]
```

7.94.3.22 operator+()

```
BTime BTime::operator+ (
            int seconds ) const  [inline]
```

7.94.3.23 operator+=()

```
BTime& BTime::operator+= (
            int seconds )  [inline]
```

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

- BTime.h
- BTime.cpp

7.95 BTimer Class Reference

Stopwatch style timer.

`#include <BTimer.h>`

Public Member Functions

- BTimer ()
- ~BTimer ()
- void start ()
  Start timer.
- void stop ()
  Stop timer.
- void clear ()
  Clear timer.
- double getElapsedTime ()
  Returns the elapsed time from the last start.
- void add (BTimer &timer)
  Add two timers.
- double average ()
  Average time is duration between start() and stop() / number of stops.
- double peak ()
  Peak time.

7.95.1 Detailed Description

Stopwatch style timer.

7.95.2 Constructor & Destructor Documentation

7.95.2.1 BTimer()

```
BTimer::BTimer ( )
```

7.95.2.2 ~BTimer()

```
BTimer::~BTimer ( )
```

7.95.3 Member Function Documentation

7.95.3.1 start()

```
void BTimer::start ( )
```

Start timer.

7.95.3.2 stop()

```
void BTimer::stop ( )
```

Stop timer.

7.95.3.3 clear()

```
void BTimer::clear ( )
```

Clear timer.

7.95.3.4 getElapsedTime()

```
double BTimer::getElapsedTime ( )
```

Returns the elapsed time from the last start.

7.95.3.5 add()

```
void BTimer::add (
            BTimer & timer )
```

Add two timers.

7.95.3.6 average()

```
double BTimer::average ( )
```

Average time is duration between start() and stop() / number of stops.

7.95.3.7 peak()

```
double BTimer::peak ( )
```

Peak time.

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

- BTimer.h
- BTimer.cpp

7.96 BTimeStamp Class Reference

A date and time storage class with microsecond resolution.

```
#include <BTimeStamp.h>
```

Public Member Functions

- BTimeStamp ()
- BTimeStamp (int year, int month=1, int day=1, int hour=0, int minute=0, int second=0, int microsecond=0)
- BTimeStamp (const BString str)
- ~BTimeStamp ()
- void clear ()

  Clear the date/time.
- void setFirst ()

  Set the first date available.
- void setLast ()

  Set the last date available.
- void set (time_t time, int microSeconds)

  Set time using Unix time (seconds from 1970-01-01)
- void set (int year=0, int month=1, int day=1, int hour=0, int minute=0, int second=0, int microsecond=0)
- void set (const BTimeStampMs &timeStamp)

  Set the timeStamp to given MS time stamp.
- void setYDay (int year=0, int yday=0, int hour=0, int minute=0, int second=0, int microsecond=0)
- void setTime (int hour=0, int minute=0, int second=0, int microsecond=0)
- void setNow ()

  Set the timeStamp to now.
- int year () const
- int yday () const
- int month () const
- int day () const
- int hour () const
- int minute () const
- int second () const
- int microSecond () const
- void getDate (int &year, int &mon, int &day) const
- BString getString (BString separator="T") const

  Get the time as an ISO date/time string.
- BError setString (const BString dateTime)

Set the time from an ISO date/time.

- BString getStringNoMs (BString separator="T") const

  Get the time as an ISO date/time string without microseconds.

- BString getStringFormatted (BString format) const

  Gets the time in a string form as per the format. Format syntax as per strftime()

- void addMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.

- void addMicroSeconds (int64_t microSeconds)

  Add the given number of micro seconds. This should be less that a year.

- void addSeconds (int seconds)

  Add the given number of seconds. This should be less that a year.

- uint32_t getYearSeconds () const

  Get number of seconds within the year.

- uint64_t getYearMicroSeconds () const

  Get number of micro seconds within the year.

- int isSet () const
- int compare (const BTimeStamp &timeStamp) const

  Compare two dates.

- operator BString () const
- BTimeStamp & operator= (const BTimeStampMs &timeStamp)
- int operator== (const BTimeStamp &timeStamp) const
- int operator!= (const BTimeStamp &timeStamp) const
- int operator> (const BTimeStamp &timeStamp) const
- int operator>= (const BTimeStamp &timeStamp) const
- int operator< (const BTimeStamp &timeStamp) const
- int operator<= (const BTimeStamp &timeStamp) const

Static Public Member Functions

- static int isLeap (int year)
- static BInt64 difference (BTimeStamp t2, BTimeStamp t1)

Public Attributes

- BUInt16 oyear

  Year (0 .. 65535)

- BUInt16 oyday

  Day in year (0 .. 365)

- BUInt8 ohour

  Hour (0 .. 23)

- BUInt8 ominute

  Minute (0 .. 59)

- BUInt8 osecond

  Second (0 .. 59)

- BUInt8 ospare

  Padding.

- BUInt32 omicroSecond

  MicroSecond (0 .. 999999)

7.96.1 Detailed Description

A date and time storage class with microsecond resolution.

7.96.2 Constructor & Destructor Documentation

7.96.2.1 BTimeStamp() [1/3]

```
BTimeStamp::BTimeStamp ( )
```

7.96.2.2 BTimeStamp() [2/3]

```
BTimeStamp::BTimeStamp (
            int year,
            int month = 1,
            int day = 1,
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.96.2.3 BTimeStamp() [3/3]

```
BTimeStamp::BTimeStamp (
            const BString str )
```

7.96.2.4 ~BTimeStamp()

```
BTimeStamp::~BTimeStamp ( )
```

7.96.3 Member Function Documentation

7.96.3.1 clear()

```
void BTimeStamp::clear ( )
```

Clear the date/time.

7.96.3.2 setFirst()

```
void BTimeStamp::setFirst ( )
```

Set the first date available.

7.96.3.3 setLast()

```
void BTimeStamp::setLast ( )
```

Set the last date available.

7.96.3.4 set() [1/3]

```
void BTimeStamp::set (
            time_t time,
            int microSeconds )
```

Set time using Unix time (seconds from 1970-01-01)

7.96.3.5 set() [2/3]

```
void BTimeStamp::set (
            int year = 0,
            int month = 1,
            int day = 1,
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.96.3.6 set() [3/3]

```
void BTimeStamp::set (
            const BTimeStampMs & timeStamp )
```

Set the timeStamp to given MS time stamp.

7.96.3.7 setYDay()

```
void BTimeStamp::setYDay (
            int year = 0,
            int yday = 0,
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.96.3.8 setTime()

```
void BTimeStamp::setTime (
            int hour = 0,
            int minute = 0,
            int second = 0,
            int microsecond = 0 )
```

7.96.3.9 setNow()

```
void BTimeStamp::setNow ( )
```

Set the timeStamp to now.

7.96.3.10 year()

```
int BTimeStamp::year ( ) const
```

7.96.3.11 yday()

```
int BTimeStamp::yday ( ) const
```

7.96.3.12 month()

```
int BTimeStamp::month ( ) const
```

7.96.3.13 day()

```
int BTimeStamp::day ( ) const
```

7.96.3.14 hour()

```
int BTimeStamp::hour ( ) const
```

7.96.3.15 minute()

```
int BTimeStamp::minute ( ) const
```

7.96.3.16 second()

```
int BTimeStamp::second ( ) const
```

7.96.3.17 microSecond()

```
int BTimeStamp::microSecond ( ) const
```

7.96.3.18 getDate()

```
void BTimeStamp::getDate (
            int & year,
            int & mon,
            int & day ) const
```

7.96.3.19 getString()

```
BString BTimeStamp::getString (
            BString separator = "T" ) const
```

Get the time as an ISO date/time string.

7.96.3.20 setString()

```
BError BTimeStamp::setString (
            const BString dateTime )
```

Set the time from an ISO date/time.

This accepts a string and parses for a supported datetime format string. The function will return an error if it cannot parse a datatime string or if one of the parsed fields is out of range. The conversion supports a fractional second resolution of 1 microsecond rounding the fractional value given to this. Supported formats include: ISO: YYYY-MM-DD[T]HH:MM:SS.FS, UK: YYYY/MM/DD[T]HH:MM:SS.FS, UK: YY/MM/DD[T]HH:MM:SS.FS, HH:MM:SS.FS and YYYYDDD[T]HH:MM:SS.FS.

7.96.3.21 getStringNoMs()

```
BString BTimeStamp::getStringNoMs (
            BString separator = "T" ) const
```

Get the time as an ISO date/time string without microseconds.

7.96.3.22 getStringFormatted()

```
BString BTimeStamp::getStringFormatted (
            BString format ) const
```

Gets the time in a string form as per the format. Format syntax as per strftime()

7.96.3.23 addMilliSeconds()

```
void BTimeStamp::addMilliSeconds (
            int milliSeconds )
```

Add the given number of milli seconds. This should be less that a year.

7.96.3.24 addMicroSeconds()

```
void BTimeStamp::addMicroSeconds (
            int64_t microSeconds )
```

Add the given number of micro seconds. This should be less that a year.

7.96.3.25 addSeconds()

```
void BTimeStamp::addSeconds (
            int seconds )
```

Add the given number of seconds. This should be less that a year.

7.96.3.26 getYearSeconds()

```
uint32_t BTimeStamp::getYearSeconds ( ) const
```

Get number of seconds within the year.

7.96.3.27 getYearMicroSeconds()

```
uint64_t BTimeStamp::getYearMicroSeconds ( ) const
```

Get number of micro seconds within the year.

7.96.3.28 isSet()

```
int BTimeStamp::isSet ( ) const  [inline]
```

7.96.3.29 compare()

```
int BTimeStamp::compare (
            const BTimeStamp & timeStamp ) const
```

Compare two dates.

7.96.3.30 operator BString()

```
BTimeStamp::operator BString ( ) const  [inline]
```

7.96.3.31 operator=()

```
BTimeStamp& BTimeStamp::operator= (
            const BTimeStampMs & timeStamp )  [inline]
```

7.96.3.32 operator==()

```
int BTimeStamp::operator== (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.33 operator"!=()

```
int BTimeStamp::operator!= (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.34 operator>()

```
int BTimeStamp::operator> (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.35 operator>=()

```
int BTimeStamp::operator>= (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.36 operator<()

```
int BTimeStamp::operator< (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.37 operator<=()

```
int BTimeStamp::operator<= (
            const BTimeStamp & timeStamp ) const  [inline]
```

7.96.3.38 isLeap()

```
int BTimeStamp::isLeap (
            int year )  [static]
```

7.96.3.39 difference()

```
BInt64 BTimeStamp::difference (
            BTimeStamp t2,
            BTimeStamp t1 )  [static]
```

7.96.4 Member Data Documentation

7.96.4.1 oyear

```
BUInt16 BTimeStamp::oyear
```

Year (0 .. 65535)

7.96.4.2 oyday

```
BUInt16 BTimeStamp::oyday
```

Day in year (0 .. 365)

7.96.4.3 ohour

```
BUInt8 BTimeStamp::ohour
```

Hour (0 .. 23)

7.96.4.4 ominute

```
BUInt8 BTimeStamp::ominute
```

Minute (0 .. 59)

7.96.4.5 osecond

```
BUInt8 BTimeStamp::osecond
```

Second (0 .. 59)

7.96.4.6 ospare

```
BUInt8 BTimeStamp::ospare
```

Padding.

7.96.4.7 omicroSecond

```
BUInt32 BTimeStamp::omicroSecond
```

MicroSecond (0 .. 999999)

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

- BTimeStamp.h
- BTimeStamp.cpp

7.97 BTimeStampMs Class Reference

A date and time storage class with milisecond resolution and an extra field to indicate a particular sampleNumber it reffers to.

```
#include <BTimeStampMs.h>
```

Public Member Functions

- BTimeStampMs (BString str="")
- ~BTimeStampMs ()
- void clear ()

  Clear the date/time.
- void setNow ()

  Set the timeStamp to now.
- void setFirst ()

  Set the first date available.
- void setLast ()

  Set the last date available.
- void set (time_t time, int milliSeconds=0)

  Set time using Unix time (seconds from 1970-01-01)
- void setYDay (int year=0, int yday=0, int hour=0, int minute=0, int second=0, int milliSecond=0)
- void setTime (int hour=0, int minute=0, int second=0, int milliSecond=0)
- BTimeStampMs & addMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.
- BTimeStampMs & subMilliSeconds (int milliSeconds)

  Add the given number of milli seconds. This should be less that a year.
- BTimeStampMs & addSeconds (int seconds)

  Add the given number of seconds. This should be less that a year.
- BTimeStampMs & subSeconds (int seconds)

  Subtract the given number of seconds. This should be less that a year.
- uint32_t getYearSeconds ()

  Get number of seconds within the year.
- uint64_t getYearMilliSeconds ()

  Get number of seconds within the year.
- BString getString (BString separator="T")

  Get the time as an ISO date/time string.
- BString getStringNoMs (BString separator="T")

  Get the time as an ISO date/time string with no ms.
- BError setString (BString dateTime)

  Set the time from an ISO date/time.
- BString getDurationString (BString separator="T")

  Get the time as an ISO date/time string but with month's and days starting from 0.
- BString getDurationStringNoMs (BString separator="T")

  Get the time as an ISO date/time string but with month's and days starting from 0 with no ms.
- BError setDurationString (BString dateTime)

  Set the time from an ISO date/time string but with month's and days starting from 0.
- BString getStringRaw ()
- void getDate (int &year, int &mon, int &day)

  Get the year, month and day.
- int compare (const BTimeStampMs &timeStamp)

  Compare two dates.
- int operator> (const BTimeStampMs &timeStamp)
- int operator>= (const BTimeStampMs &timeStamp)
- int operator< (const BTimeStampMs &timeStamp)
- int operator<= (const BTimeStampMs &timeStamp)

Static Public Member Functions

- static int isLeap (int year)
- static BUInt64 difference (BTimeStampMs t2, BTimeStampMs t1)

Public Attributes

- uint16_t year

  Year (2000 .. 3000)
- uint16_t yday

  Day in year (0 .. 365)
- uint16_t hour

  Hour (0 .. 23)
- uint16_t minute

  Minute (0 .. 59)
- uint16_t second

  Second (0 .. 59)
- uint16_t milliSecond

  MilliSecond (0 .. 999)
- int32_t sampleNumber

  The sample number this time refers to.

7.97.1 Detailed Description

A date and time storage class with milisecond resolution and an extra field to indicate a particular sampleNumber it reffers to.

7.97.2 Constructor & Destructor Documentation

7.97.2.1 BTimeStampMs()

```
BTimeStampMs::BTimeStampMs (
            BString str = "" )
```

7.97.2.2 ~BTimeStampMs()

```
BTimeStampMs::~BTimeStampMs ( )
```

7.97.3 Member Function Documentation

7.97.3.1 clear()

```
void BTimeStampMs::clear ( )
```

Clear the date/time.

7.97.3.2 setNow()

```
void BTimeStampMs::setNow ( )
```

Set the timeStamp to now.

7.97.3.3 setFirst()

```
void BTimeStampMs::setFirst ( )
```

Set the first date available.

7.97.3.4 setLast()

```
void BTimeStampMs::setLast ( )
```

Set the last date available.

7.97.3.5 set()

```
void BTimeStampMs::set (
            time_t time,
            int milliSeconds = 0 )
```

Set time using Unix time (seconds from 1970-01-01)

7.97.3.6 setYDay()

```
void BTimeStampMs::setYDay (
            int year = 0,
            int yday = 0,
            int hour = 0,
            int minute = 0,
            int second = 0,
            int milliSecond = 0 )
```

7.97.3.7 setTime()

```
void BTimeStampMs::setTime (
             int hour = 0,
             int minute = 0,
             int second = 0,
             int milliSecond = 0 )
```

7.97.3.8 addMilliSeconds()

```
BTimeStampMs & BTimeStampMs::addMilliSeconds (
             int milliSeconds )
```

Add the given number of milli seconds. This should be less that a year.

7.97.3.9 subMilliSeconds()

```
BTimeStampMs & BTimeStampMs::subMilliSeconds (
             int milliSeconds )
```

Add the given number of milli seconds. This should be less that a year.

7.97.3.10 addSeconds()

```
BTimeStampMs & BTimeStampMs::addSeconds (
             int seconds )
```

Add the given number of seconds. This should be less that a year.

7.97.3.11 subSeconds()

```
BTimeStampMs & BTimeStampMs::subSeconds (
             int seconds )
```

Subtract the given number of seconds. This should be less that a year.

7.97.3.12 getYearSeconds()

```
uint32_t BTimeStampMs::getYearSeconds ( )
```

Get number of seconds within the year.

7.97.3.13 getYearMilliSeconds()

```
uint64_t BTimeStampMs::getYearMilliSeconds ( )
```

Get number of seconds within the year.

7.97.3.14 getString()

```
BString BTimeStampMs::getString (
            BString separator = "T" )
```

Get the time as an ISO date/time string.

7.97.3.15 getStringNoMs()

```
BString BTimeStampMs::getStringNoMs (
            BString separator = "T" )
```

Get the time as an ISO date/time string with no ms.

7.97.3.16 setString()

```
BError BTimeStampMs::setString (
            BString dateTime )
```

Set the time from an ISO date/time.

This accepts a string and parses for a supported datetime format string. The function will return an error if it cannot parse a datatime string or if one of the parsed fields is out of range. The conversion supports a fractional second resolution of 1 milisecond resolution rounding the fractional value given to this. Supported formats include: ISO: YYYY-MM-DD[T]HH:MM:SS.FS.

7.97.3.17 getDurationString()

```
BString BTimeStampMs::getDurationString (
            BString separator = "T" )
```

Get the time as an ISO date/time string but with month's and days starting from 0.

7.97.3.18 getDurationStringNoMs()

```
BString BTimeStampMs::getDurationStringNoMs (
            BString separator = "T" )
```

Get the time as an ISO date/time string but with month's and days starting from 0 with no ms.

7.97.3.19 setDurationString()

```
BError BTimeStampMs::setDurationString (
            BString dateTime )
```

Set the time from an ISO date/time string but with month's and days starting from 0.

7.97.3.20 getStringRaw()

```
BString BTimeStampMs::getStringRaw ( )
```

7.97.3.21 getDate()

```
void BTimeStampMs::getDate (
            int & year,
            int & mon,
            int & day )
```

Get the year, month and day.

7.97.3.22 compare()

```
int BTimeStampMs::compare (
            const BTimeStampMs & timeStamp )
```

Compare two dates.

7.97.3.23 operator>()

```
int BTimeStampMs::operator> (
            const BTimeStampMs & timeStamp )  [inline]
```

7.97.3.24 operator>=()

```
int BTimeStampMs::operator>= (
            const BTimeStampMs & timeStamp )  [inline]
```

7.97.3.25 operator<()

```
int BTimeStampMs::operator< (
            const BTimeStampMs & timeStamp )  [inline]
```

7.97.3.26 operator<=()

```
int BTimeStampMs::operator<= (
            const BTimeStampMs & timeStamp )  [inline]
```

7.97.3.27 isLeap()

```
int BTimeStampMs::isLeap (
            int year )  [static]
```

7.97.3.28 difference()

```
BUInt64 BTimeStampMs::difference (
            BTimeStampMs t2,
            BTimeStampMs t1 )  [static]
```

7.97.4 Member Data Documentation

7.97.4.1 year

```
uint16_t BTimeStampMs::year
```

Year (2000 .. 3000)

7.97.4.2 yday

```
uint16_t BTimeStampMs::yday
```

Day in year (0 .. 365)

7.97.4.3 hour

```
uint16_t BTimeStampMs::hour
```

Hour (0 .. 23)

7.97.4.4 minute

```
uint16_t BTimeStampMs::minute
```

Minute (0 .. 59)

7.97.4.5 second

```
uint16_t BTimeStampMs::second
```

Second (0 .. 59)

7.97.4.6 milliSecond

```
uint16_t BTimeStampMs::milliSecond
```

MilliSecond (0 .. 999)

7.97.4.7 sampleNumber

```
int32_t BTimeStampMs::sampleNumber
```

The sample number this time refers to.

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

- BTimeStampMs.h
- BTimeStampMs.cpp

7.98 BTimeUs Class Reference

Time storage as an unsigned 64bit value to TAI standard.

```
#include <BTimeUs.h>
```

Public Member Functions

- BTimeUs (BUInt64 t=0)
- BTimeUs (BTime t)
- void set (BUInt64 microSeconds)

  Set the time to TAI us.
- void set (BUInt year, BUInt month, BUInt day, BUInt hour=0, BUInt minute=0, BUInt second=0, BUInt microSecond=0)

  Set the date and time from UTC.
- void setYearDay (BUInt year, BUInt yearDay, BUInt hour=0, BUInt minute=0, BUInt second=0, BUInt microSecond=0)

  Set the date and time from UTC.
- void getDate (BUInt &year, BUInt &month, BUInt &day) const

  Return the date information UTC.
- void getTime (BUInt &hour, BUInt &minute, BUInt &second) const

  Return the time information UTC.
- BUInt64 getSeconds () const

  Return the number of seconds TAI.
- BUInt64 getMicroSeconds () const

  Return the number of micro seconds TAI.
- int isSet () const

  Check if set.
- int isLeapYear ()

  Returns if a leap year.
- void addSeconds (BInt64 seconds)

  Add the given number of seconds.
- void addMicroSeconds (BInt64 microSeconds)

  Add the given number of seconds.
- BString getString (BString format="isoT") const

  Gets the date/time in string format.
- BString getStringUs (BString format="isoT") const

  Gets the date/time in string format.

- BError setString (const BString dateTime)

  Sets the date/time from string format.
- operator BTime () const
- int operator== (const BTimeUs &time) const
- int operator!= (const BTimeUs &time) const
- int operator> (const BTimeUs &time) const
- int operator>= (const BTimeUs &time) const
- int operator< (const BTimeUs &time) const
- int operator<= (const BTimeUs &time) const
- BTimeUs operator+ (BInt64 microSeconds) const
- BTimeUs & operator+= (BInt64 microSeconds)

7.98.1 Detailed Description

Time storage as an unsigned 64bit value to TAI standard.

7.98.2 Constructor & Destructor Documentation

7.98.2.1 BTimeUs() [1/2]

```
BTimeUs::BTimeUs (
            BUInt64 t = 0 )
```

7.98.2.2 BTimeUs() [2/2]

```
BTimeUs::BTimeUs (
            BTime t )
```

7.98.3 Member Function Documentation

7.98.3.1 set() [1/2]

```
void BTimeUs::set (
            BUInt64 microSeconds )
```

Set the time to TAI us.

7.98.3.2 set() [2/2]

```
void BTimeUs::set (
            BUInt year,
            BUInt month,
            BUInt day,
            BUInt hour = 0,
            BUInt minute = 0,
            BUInt second = 0,
            BUInt microSecond = 0 )
```

Set the date and time from UTC.

7.98.3.3 setYearDay()

```
void BTimeUs::setYearDay (
            BUInt year,
            BUInt yearDay,
            BUInt hour = 0,
            BUInt minute = 0,
            BUInt second = 0,
            BUInt microSecond = 0 )
```

Set the date and time from UTC.

7.98.3.4 getDate()

```
void BTimeUs::getDate (
            BUInt & year,
            BUInt & month,
            BUInt & day ) const
```

Return the date information UTC.

7.98.3.5 getTime()

```
void BTimeUs::getTime (
            BUInt & hour,
            BUInt & minute,
            BUInt & second ) const
```

Return the time information UTC.

7.98.3.6 getSeconds()

```
BUInt64 BTimeUs::getSeconds ( ) const
```

Return the number of seconds TAI.

7.98.3.7 getMicroSeconds()

```
BUInt64 BTimeUs::getMicroSeconds ( ) const
```

Return the number of micro seconds TAI.

7.98.3.8 isSet()

```
int BTimeUs::isSet ( ) const  [inline]
```

Check if set.

7.98.3.9 isLeapYear()

```
int BTimeUs::isLeapYear ( )
```

Returns if a leap year.

7.98.3.10 addSeconds()

```
void BTimeUs::addSeconds (
            BInt64 seconds )
```

Add the given number of seconds.

7.98.3.11 addMicroSeconds()

```
void BTimeUs::addMicroSeconds (
            BInt64 microSeconds )
```

Add the given number of seconds.

7.98.3.12 getString()

```
BString BTimeUs::getString (
            BString format = "isoT" ) const
```

Gets the date/time in string format.

7.98.3.13 getStringUs()

```
BString BTimeUs::getStringUs (
            BString format = "isoT" ) const
```

Gets the date/time in string format.

7.98.3.14 setString()

```
BError BTimeUs::setString (
            const BString dateTime )
```

Sets the date/time from string format.

7.98.3.15 operator BTime()

```
BTimeUs::operator BTime ( ) const  [inline]
```

7.98.3.16 operator==()

```
int BTimeUs::operator== (
            const BTimeUs & time ) const  [inline]
```

7.98.3.17 operator"!=()

```
int BTimeUs::operator!= (
            const BTimeUs & time ) const  [inline]
```

7.98.3.18 operator>()

```
int BTimeUs::operator> (
            const BTimeUs & time ) const  [inline]
```

7.98.3.19 operator>=()

```
int BTimeUs::operator>= (
            const BTimeUs & time ) const  [inline]
```

7.98.3.20 operator<()

```
int BTimeUs::operator< (
            const BTimeUs & time ) const  [inline]
```

7.98.3.21 operator<=()

```
int BTimeUs::operator<= (
            const BTimeUs & time ) const  [inline]
```

7.98.3.22 operator+()

```
BTimeUs BTimeUs::operator+ (
            BInt64 microSeconds ) const  [inline]
```

7.98.3.23 operator+=()

```
BTimeUs& BTimeUs::operator+= (
            BInt64 microSeconds )  [inline]
```

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

- BTimeUs.h
- BTimeUs.cpp

7.99 BUrl Class Reference

Access to a Url.

`#include <BUrl.h>`

Public Member Functions

- BUrl ()
- ~BUrl ()
- BError readString (BString url, BString &str)

  Reads URL.

7.99.1 Detailed Description

Access to a Url.

7.99.2 Constructor & Destructor Documentation

7.99.2.1 BUrl()

```
BUrl::BUrl ( )
```

7.99.2.2 ~BUrl()

```
BUrl::~BUrl ( )
```

7.99.3 Member Function Documentation

7.99.3.1 readString()

```
BError BUrl::readString (
            BString url,
            BString & str )
```

Reads URL.

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

- BUrl.h
- BUrl.cpp

Chapter 8

File Documentation

8.1 BArray.h File Reference

```
#include <BTypes.h>
#include <vector>
#include <algorithm>
```

Classes

- class BArray< T >

  Template based Array class.

Macros

- #define BArrayLoop(list, i) for(BUInt i = 0; i < list.number(); i++)

8.1.1 Macro Definition Documentation

8.1.1.1 BArrayLoop

```
#define BArrayLoop(
             list,
             i ) for(BUInt i = 0; i < list.number(); i++)
```

8.2 BAtomic.h File Reference

```
#include <BTypes.h>
```

Classes

- class BAtomic< Type >

  BAtomic class increments/decrements different integer types.

Typedefs

- typedef BAtomic< BInt32 > BAtomicInt32
- typedef BAtomic< BInt64 > BAtomicInt64
- typedef BAtomic< BUInt32 > BAtomicUInt32
- typedef BAtomic< BUInt64 > BAtomicUInt64

8.2.1 Typedef Documentation

8.2.1.1 BAtomicInt32

```
typedef BAtomic<BInt32> BAtomicInt32
```

8.2.1.2 BAtomicInt64

```
typedef BAtomic<BInt64> BAtomicInt64
```

8.2.1.3 BAtomicUInt32

```
typedef BAtomic<BUInt32> BAtomicUInt32
```

8.2.1.4 BAtomicUInt64

```
typedef BAtomic<BUInt64> BAtomicUInt64
```

8.3 BAtomicCount.h File Reference

```
#include <bits/atomicity.h>
```

Classes

- class BAtomicCount

  BAtomicCount class.

8.4 BBuffer.cpp File Reference

```
#include <stdlib.h>
#include <memory.h>
#include <BBuffer.h>
#include <BEndian.h>
#include <BTimeStamp.h>
#include <BComplex.h>
```

Variables

- const int roundSize = 256

8.4.1 Variable Documentation

8.4.1.1 roundSize

```
const int roundSize = 256
```

8.5 BBuffer.h File Reference

```
#include <BTypes.h>
#include <BString.h>
#include <BError.h>
#include <BComplex.h>
#include <BEndian.h>
```

Classes

- class BBuffer

  Create and manipulate a variable sized byte data buffer.

- class BBufferStore

  Create and manipulate a variable sized byte data buffer. Has functions to store and retrieve basic and extended types/classes in the binary buffer.

Macros

- #define BBigEndian 0

8.5.1 Macro Definition Documentation

8.5.1.1 BBigEndian

```
#define BBigEndian 0
```

8.6 BComms.cpp File Reference

```
#include <BComms.h>
```

8.7 BComms.h File Reference

```
#include <BTypes.h>
#include <BEvent.h>
#include <BError.h>
```

Classes

- class BComms

  A base class for communications classes having a generic API.

8.8 BComplex.h File Reference

```
#include <BTypes.h>
#include <complex>
#include <algorithm>
```

Typedefs

- typedef std::complex< BFloat64 > BComplex

  This is a complex number using BFloat64 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.
- typedef std::complex< BFloat32 > BComplex32

  This is a complex number using BFloat32 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.
- typedef std::complex< BFloat64 > BComplex64

  This is a complex number using BFloat64 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.

8.8.1 Typedef Documentation

8.8.1.1 BComplex

```
typedef std::complex<BFloat64> BComplex
```

This is a complex number using BFloat64 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.

8.8.1.2 BComplex32

```
typedef std::complex<BFloat32> BComplex32
```

This is a complex number using BFloat32 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.

8.8.1.3 BComplex64

```
typedef std::complex<BFloat64> BComplex64
```

This is a complex number using BFloat64 sized parameters. It is based on the Standard C++ library complex class and has all of the functionality of that class.

8.9 BCond.cpp File Reference

```
#include <BCond.h>
#include <sys/time.h>
#include <stdio.h>
```

8.10 BCond.h File Reference

```
#include <pthread.h>
```

Classes

- class BCond

  Thread safe conditional variable.

8.11 BCondInt.cpp File Reference

```
#include <BCondInt.h>
#include <sys/time.h>
#include <stdio.h>
#include <errno.h>
```

Functions

- static struct timespec getTimeout (uint32_t timeOutUs)

8.11.1 Function Documentation

8.11.1.1 getTimeout()

```
static struct timespec getTimeout (
            uint32_t timeOutUs )  [static]
```

8.12 BCondInt.h File Reference

```
#include <BTypes.h>
#include <pthread.h>
```

Classes

- class BCondInt

  Thread conditional value.
- class BCondValue

  Thread conditional value.
- class BCondBool

  Thread conditional boolean.
- class BCondWrap

  Thread conditional unsigned 32 bit integer value that can wrap around.
- class BCondResource

  Resource lock.

8.13 BConfig.cpp File Reference

```
#include <BConfig.h>
#include <string.h>
```

8.14 BConfig.h File Reference

```
#include <BDict.h>
#include <BFile.h>
#include <BMutex.h>
```

Classes

- class BConfig

  This class implements the configuration file access.

8.15 BCrc16.cpp File Reference

```
#include <BCrc16.h>
```

Functions

- BUInt16 bcrc16 (void *buf, BUInt16 len)

  A 16bit CRC generator.

Variables

- static const BUInt8 table_crc_hi [ ]
- static const BUInt8 table_crc_lo [ ]

8.15.1 Function Documentation

8.15.1.1 bcrc16()

```
BUInt16 bcrc16 (
            void * buf,
            BUInt16 len )
```

A 16bit CRC generator.

8.15.2 Variable Documentation

8.15.2.1 table_crc_hi

```
const BUInt8 table_crc_hi[]  [static]
```

Initial value:

```
= {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
}
```

8.15.2.2 table_crc_lo

```
const BUInt8 table_crc_lo[]  [static]
```

Initial value:

```
= {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
    0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
    0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
    0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
    0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
    0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
    0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
    0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
    0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
    0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
    0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
    0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
    0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
    0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
    0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
    0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
    0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
    0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
    0x43, 0x83, 0x41, 0x81, 0x80, 0x40
}
```

8.16 BCrc16.h File Reference

```
#include <BTypes.h>
```

Functions

- BUInt16 bcrc16 (void *buf, BUInt16 len)
  A 16bit CRC generator.

8.16.1 Function Documentation

8.16.1.1 bcrc16()

```
BUInt16 bcrc16 (
            void * buf,
            BUInt16 len )
```

A 16bit CRC generator.

8.17 BCrc32.cpp File Reference

```
#include <BCrc32.h>
```

Functions

- BUInt32 bcrc32 (BUInt32 crc, const void *buf, BUInt32 len)
  A 32bit CRC generator.

Variables

- static BUInt32 crc32_tab [ ]

8.17.1 Function Documentation

8.17.1.1 bcrc32()

```
BUInt32 bcrc32 (
            BUInt32 crc,
            const void * buf,
            BUInt32 len )
```

A 32bit CRC generator.

8.17.2 Variable Documentation

8.17.2.1 crc32_tab

```
BUInt32 crc32_tab[]  [static]
```

8.18 BCrc32.h File Reference

```
#include <BTypes.h>
```

Functions

- BUInt32 bcrc32 (BUInt32 crc, const void *buf, BUInt32 len)
  A 32bit CRC generator.

8.18.1 Function Documentation

8.18.1.1 bcrc32()

```
BUInt32 bcrc32 (
            BUInt32 crc,
            const void * buf,
            BUInt32 len )
```

A 32bit CRC generator.

8.19 BDate.cpp File Reference

```
#include <BDate.h>
#include <sys/time.h>
```

Functions

- void toBString (BDate &v, BString &s)
- void fromBString (BString &s, BDate &v)

Variables

- static int mon_yday [2][13]

8.19.1 Function Documentation

8.19.1.1 toBString()

```
void toBString (
            BDate & v,
            BString & s )
```

8.19.1.2 fromBString()

```
void fromBString (
            BString & s,
            BDate & v )
```

8.19.2 Variable Documentation

8.19.2.1 mon_yday

```
int mon_yday[2][13]  [static]
```

Initial value:

```
= {
    { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
    { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.20 BDate.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BDate

  This class store a UTC calendar date as a year and a year's day.

Functions

- void toBString (BDate &v, BString &s)
- void fromBString (BString &s, BDate &v)

8.20.1 Function Documentation

8.20.1.1 toBString()

```
void toBString (
            BDate & v,
            BString & s )
```

8.20.1.2 fromBString()

```
void fromBString (
            BString & s,
            BDate & v )
```

8.21 BDebug.cpp File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <stdarg.h>
#include <fcntl.h>
#include <ctype.h>
#include <BDebug.h>
```

Functions

- void bhd8 (const void *data, unsigned int n)

  Software debug functions.
- void bhd8a (const void *data, unsigned int n)
- void bhda8 (const void *data, unsigned int n)
- void bhd32 (const void *data, unsigned int n)
- void bhda32 (const void *data, unsigned int n)
- double getTime ()
- void setDebug (int d)
- void tprintf (int log, const char *fmt,...)

Variables

- int bdebug

8.21.1 Function Documentation

8.21.1.1 bhd8()

```
void bhd8 (
            const void * data,
            unsigned int n )
```

Software debug functions.

8.21.1.2 bhd8a()

```
void bhd8a (
            const void * data,
            unsigned int n )
```

8.21.1.3 bhda8()

```
void bhda8 (
            const void * data,
            unsigned int n )
```

8.21.1.4 bhd32()

```
void bhd32 (
            const void * data,
            unsigned int n )
```

8.21.1.5 bhda32()

```
void bhda32 (
            const void * data,
            unsigned int n )
```

8.21.1.6 getTime()

```
double getTime ( )
```

8.21.1.7 setDebug()

```
void setDebug (
            int d )
```

8.21.1.8 tprintf()

```
void tprintf (
            int log,
            const char * fmt,
             ...  )
```

8.21.2 Variable Documentation

8.21.2.1 bdebug

```
int bdebug
```

8.22 BDebug.h File Reference

```
#include <stdio.h>
#include <time.h>
#include <syslog.h>
```

Classes

- class BDebugBacktrace

  Backtrace on crash class.

Macros

- #define BDebug_STD 0x000001
- #define dprintf(level, fmt, a...)

  General debug functions.
- #define nprintf(fmt, a...) syslog(LOG_NOTICE, fmt, ##a)

  Warnings and errors logging.
- #define wprintf(fmt, a...) syslog(LOG_WARNING, fmt, ##a)
- #define eprintf(fmt, a...) syslog(LOG_ERR, fmt, ##a)
- #define dl1printf(fmt, a...)
- #define dl2printf(fmt, a...)
- #define dl3printf(fmt, a...)
- #define dl4printf(fmt, a...)

Functions

- void bhd8 (const void *data, unsigned int n)

  Software debug functions.
- void bhd8a (const void *data, unsigned int n)
- void bhda8 (const void *data, unsigned int n)
- void bhd32 (const void *data, unsigned int n)
- void bhds32 (const void *data, unsigned int n)
- double getTime ()
- void setDebug (int debug)
- void tprintf (int log, const char *fmt,...)
- pid_t bgettid ()

Variables

- int bdebug

8.22.1 Macro Definition Documentation

8.22.1.1 BDebug_STD

```
#define BDebug_STD 0x000001
```

8.22.1.2 dprintf

```
#define dprintf(
              level,
              fmt,
              a...  )
```

General debug functions.

8.22.1.3 nprintf

```
#define nprintf(
              fmt,
              a...  ) syslog(LOG_NOTICE, fmt, ##a)
```

Warnings and errors logging.

8.22.1.4 wprintf

```
#define wprintf(
              fmt,
              a...  ) syslog(LOG_WARNING, fmt, ##a)
```

8.22.1.5 eprintf

```
#define eprintf(
              fmt,
              a...  ) syslog(LOG_ERR, fmt, ##a)
```

8.22.1.6 dl1printf

```
#define dl1printf(
              fmt,
              a...  )
```

8.22.1.7 dl2printf

```
#define dl2printf(
              fmt,
              a...  )
```

8.22.1.8 dl3printf

```
#define dl3printf(
              fmt,
              a...  )
```

8.22.1.9 dl4printf

```
#define dl4printf(
              fmt,
              a...  )
```

8.22.2 Function Documentation

8.22.2.1 bhd8()

```
void bhd8 (
             const void * data,
             unsigned int n )
```

Software debug functions.

8.22.2.2 bhd8a()

```
void bhd8a (
             const void * data,
             unsigned int n )
```

8.22.2.3 bhda8()

```
void bhda8 (
             const void * data,
             unsigned int n )
```

8.22.2.4 bhd32()

```
void bhd32 (
             const void * data,
             unsigned int n )
```

8.22.2.5 bhds32()

```
void bhds32 (
             const void * data,
             unsigned int n )
```

8.22.2.6 getTime()

```
double getTime ( )
```

8.22.2.7 setDebug()

```
void setDebug (
             int debug )
```

8.22.2.8 tprintf()

```
void tprintf (
             int log,
             const char * fmt,
              ...  )
```

8.22.2.9 bgettid()

```
pid_t bgettid ( )
```

8.22.3 Variable Documentation

8.22.3.1 bdebug

```
int bdebug  [extern]
```

8.23 BDict.cpp File Reference

```
#include <BDict.h>
```

Functions

- void toBString (const BDictString &v, BString &s)
- void fromBString (const BString &str, BDictString &v)
- BString bdictStringToString (const BDictString &dict)

8.23.1 Function Documentation

8.23.1.1 toBString()

```
void toBString (
            const BDictString & v,
            BString & s )
```

8.23.1.2 fromBString()

```
void fromBString (
            const BString & str,
            BDictString & v )
```

8.23.1.3 bdictStringToString()

```
BString bdictStringToString (
            const BDictString & dict )
```

8.24 BDict.h File Reference

```
#include <BNameValue.h>
```

Classes

- class BDictItem< Type >

  Template based Dictionary classes item.
- class BDict< Type >

  Dictionary list class using templates.

Typedefs

- typedef BDict< BString > BDictString

Functions

- void toBString (const BDictString &v, BString &s)
- void fromBString (const BString &s, BDictString &v)
- BString bdictStringToString (const BDictString &dict)

8.24.1 Typedef Documentation

8.24.1.1 BDictString

```
typedef BDict<BString> BDictString
```

8.24.2 Function Documentation

8.24.2.1 toBString()

```
void toBString (
            const BDictString & v,
            BString & s )
```

8.24.2.2 fromBString()

```
void fromBString (
            const BString & s,
            BDictString & v )
```

8.24.2.3 bdictStringToString()

```
BString bdictStringToString (
            const BDictString & dict )
```

8.25 BDictMap.h File Reference

```
#include <BString.h>
#include <map>
```

Classes

- class BDictMap< Value >
  Mapped Dictionary class.

Typedefs

- typedef BDictMap< BString > BDictMapString

8.25.1 Typedef Documentation

8.25.1.1 BDictMapString

```
typedef BDictMap<BString> BDictMapString
```

8.26 BDir.cpp File Reference

```
#include <BDir.h>
#include <dirent.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
```

Functions

- static int wild (const dirent *e)

Variables

- static BString wildString

8.26.1 Function Documentation

8.26.1.1 wild()

```
static int wild (
            const dirent * e )  [static]
```

8.26.2 Variable Documentation

8.26.2.1 wildString

```
BString wildString  [static]
```

8.27 BDir.h File Reference

```
#include <BList.h>
#include <BString.h>
#include <BError.h>
#include <sys/stat.h>
```

Classes

- class BDir

  File system directory class.

8.28 BDuration.cpp File Reference

```
#include <BDuration.h>
#include <sys/time.h>
```

8.29 BDuration.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BDuration

  Stores and manipulates a time to the nearest microsecond and a maximum of 24 hours.

8.30 BEndian.cpp File Reference

```
#include <BEndian.h>
#include <memory.h>
```

Functions

- void bswap_copy (int swap, const void *src, void *dst, BUInt32 nBytes, const char *swapType)

8.30.1 Function Documentation

8.30.1.1 bswap_copy()

```
void bswap_copy (
            int swap,
            const void * src,
            void * dst,
            BUInt32 nBytes,
            const char * swapType )
```

8.31 BEndian.h File Reference

```
#include <BTypes.h>
#include <byteswap.h>
```

Macros

- #define htobe16(x) __bswap_16 (x)
- #define htole16(x) (x)
- #define be16toh(x) __bswap_16 (x)
- #define le16toh(x) (x)
- #define htobe32(x) __bswap_32 (x)
- #define htole32(x) (x)
- #define be32toh(x) __bswap_32 (x)
- #define le32toh(x) (x)
- #define htobe64(x) __bswap_64 (x)
- #define htole64(x) (x)
- #define be64toh(x) __bswap_64 (x)
- #define le64toh(x) (x)

Functions

- void bswap_p8 (const void *s, void *d)
- void bswap_p16 (const void *s, void *d)
- void bswap_p32 (const void *s, void *d)
- void bswap_p64 (const void *s, void *d)
- void bswap_copy (int swap, const void *src, void *dst, BUInt32 nBytes, const char *swapType)
- uint16_t htole (uint16_t v)
- int16_t htole (int16_t v)
- uint32_t htole (uint32_t v)
- int32_t htole (int32_t v)
- uint64_t htole (uint64_t v)
- int64_t htole (int64_t v)
- double htole (double v)
- float htole (float v)
- uint16_t htobe (uint16_t v)
- int16_t htobe (int16_t v)
- uint32_t htobe (uint32_t v)
- int32_t htobe (int32_t v)
- uint64_t htobe (uint64_t v)
- int64_t htobe (int64_t v)
- double htobe (double v)
- float htobe (float v)
- uint16_t letoh (uint16_t v)
- int16_t letoh (int16_t v)
- uint32_t letoh (uint32_t v)
- int32_t letoh (int32_t v)
- uint64_t letoh (uint64_t v)
- int64_t letoh (int64_t v)
- double letoh (double v)
- float letoh (float v)
- uint16_t betoh (uint16_t v)
- int16_t betoh (int16_t v)
- uint32_t betoh (uint32_t v)
- int32_t betoh (int32_t v)
- uint64_t betoh (uint64_t v)
- int64_t betoh (int64_t v)
- double betoh (double v)
- float betoh (float v)

8.31.1 Macro Definition Documentation

8.31.1.1 htobe16

```
#define htobe16(
              x ) __bswap_16 (x)
```

8.31.1.2 htole16

```
#define htole16(
              x ) (x)
```

8.31.1.3 be16toh

```
#define be16toh(
              x ) __bswap_16 (x)
```

8.31.1.4 le16toh

```
#define le16toh(
              x ) (x)
```

8.31.1.5 htobe32

```
#define htobe32(
              x ) __bswap_32 (x)
```

8.31.1.6 htole32

```
#define htole32(
              x ) (x)
```

8.31.1.7 be32toh

```
#define be32toh(
              x ) __bswap_32 (x)
```

8.31.1.8 le32toh

```
#define le32toh(
              x ) (x)
```

8.31.1.9 htobe64

```
#define htobe64(
              x ) __bswap_64 (x)
```

8.31.1.10 htole64

```
#define htole64(
              x ) (x)
```

8.31.1.11 be64toh

```
#define be64toh(
              x ) __bswap_64 (x)
```

8.31.1.12 le64toh

```
#define le64toh(
              x ) (x)
```

8.31.2 Function Documentation

8.31.2.1 bswap_p8()

```
void bswap_p8 (
             const void * s,
             void * d )  [inline]
```

8.31.2.2 bswap_p16()

```
void bswap_p16 (
             const void * s,
             void * d )  [inline]
```

8.31.2.3 bswap_p32()

```
void bswap_p32 (
            const void * s,
            void * d )  [inline]
```

8.31.2.4 bswap_p64()

```
void bswap_p64 (
            const void * s,
            void * d )  [inline]
```

8.31.2.5 bswap_copy()

```
void bswap_copy (
            int swap,
            const void * src,
            void * dst,
            BUInt32 nBytes,
            const char * swapType )
```

8.31.2.6 htole() [1/8]

```
uint16_t htole (
            uint16_t v )  [inline]
```

8.31.2.7 htole() [2/8]

```
int16_t htole (
            int16_t v )  [inline]
```

8.31.2.8 htole() [3/8]

```
uint32_t htole (
            uint32_t v )  [inline]
```

8.31.2.9 htole() [4/8]

```
int32_t htole (
            int32_t v )  [inline]
```

8.31.2.10 htole() [5/8]

```
uint64_t htole (
            uint64_t v )  [inline]
```

8.31.2.11 htole() [6/8]

```
int64_t htole (
            int64_t v )  [inline]
```

8.31.2.12 htole() [7/8]

```
double htole (
            double v )  [inline]
```

8.31.2.13 htole() [8/8]

```
float htole (
            float v )  [inline]
```

8.31.2.14 htobe() [1/8]

```
uint16_t htobe (
            uint16_t v )  [inline]
```

8.31.2.15 htobe() [2/8]

```
int16_t htobe (
            int16_t v )  [inline]
```

8.31.2.16 htobe() [3/8]

```
uint32_t htobe (
            uint32_t v )  [inline]
```

8.31.2.17 htobe() [4/8]

```
int32_t htobe (
            int32_t v )  [inline]
```

8.31.2.18 htobe() [5/8]

```
uint64_t htobe (
            uint64_t v )  [inline]
```

8.31.2.19 htobe() [6/8]

```
int64_t htobe (
            int64_t v )  [inline]
```

8.31.2.20 htobe() [7/8]

```
double htobe (
            double v )  [inline]
```

8.31.2.21 htobe() [8/8]

```
float htobe (
            float v )  [inline]
```

8.31.2.22 letoh() [1/8]

```
uint16_t letoh (
            uint16_t v )  [inline]
```

8.31.2.23 letoh() [2/8]

```
int16_t letoh (
            int16_t v )  [inline]
```

8.31.2.24 letoh() [3/8]

```
uint32_t letoh (
            uint32_t v )  [inline]
```

8.31.2.25 letoh() [4/8]

```
int32_t letoh (
            int32_t v )  [inline]
```

8.31.2.26 letoh() [5/8]

```
uint64_t letoh (
            uint64_t v )  [inline]
```

8.31.2.27 letoh() [6/8]

```
int64_t letoh (
            int64_t v )  [inline]
```

8.31.2.28 letoh() [7/8]

```
double letoh (
            double v )  [inline]
```

8.31.2.29 letoh() [8/8]

```
float letoh (
            float v )  [inline]
```

8.31.2.30 betoh() [1/8]

```
uint16_t betoh (
            uint16_t v )  [inline]
```

8.31.2.31 betoh() [2/8]

```
int16_t betoh (
            int16_t v )  [inline]
```

8.31.2.32 betoh() [3/8]

```
uint32_t betoh (
            uint32_t v )  [inline]
```

8.31.2.33 betoh() [4/8]

```
int32_t betoh (
            int32_t v )  [inline]
```

8.31.2.34 betoh() [5/8]

```
uint64_t betoh (
            uint64_t v )  [inline]
```

8.31.2.35 betoh() [6/8]

```
int64_t betoh (
            int64_t v )  [inline]
```

8.31.2.36 betoh() [7/8]

```
double betoh (
            double v )  [inline]
```

8.31.2.37 betoh() [8/8]

```
float betoh (
            float v )  [inline]
```

8.32 BEntry.cpp File Reference

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <BEntry.h>
```

8.33 BEntry.h File Reference

```
#include <BList.h>
#include <BString.h>
```

Classes

- class BEntry

  Manipulate a name value pair.
- class BEntryList

  List of Entries. Where each entry is a name value pair.
- class BEntryFile

  A file based list of string name/value pairs.

8.34 BError.cpp File Reference

```
#include <BError.h>
```

8.35 BError.h File Reference

```
#include <BString.h>
```

Classes

- class BError

  Error return class. This class is used to return the error status from a function. It encapsulates an integer error number and a string.

Enumerations

- enum BErrorNum {
  ErrorOk = 0 , ErrorMisc = 1 , ErrorWarning = 2 , ErrorParam = 3 ,
  ErrorTimeout = 4 , ErrorNotAvailable = 5 , ErrorData = 6 , ErrorChecksum = 7 ,
  ErrorOverrun = 8 , ErrorUnderrun = 9 , ErrorInit = 10 , ErrorConfig = 11 ,
  ErrorNotImplemented = 12 , ErrorResourceLimit = 13 , ErrorEndOfFile = 14 , ErrorFile = 15 ,
  ErrorFormat = 16 , ErrorComms = 17 , ErrorAccessDenied = 18 , ErrorNoData = 19 ,
  ErrorEndOfData = 20 , ErrorDataPresent = 21 , ErrorDataTruncated = 22 , ErrorApiVersion = 23 ,
  ErrorAppBase = 64 , ErrorUserBase = 96 }

8.35.1 Enumeration Type Documentation

8.35.1.1 BErrorNum

`enum BErrorNum`

Enumerator

ErrorOk	
ErrorMisc	
ErrorWarning	
ErrorParam	
ErrorTimeout	
ErrorNotAvailable	
ErrorData	
ErrorChecksum	
ErrorOverrun	
ErrorUnderrun	
ErrorInit	
ErrorConfig	
ErrorNotImplemented	
ErrorResourceLimit	
ErrorEndOfFile	
ErrorFile	
ErrorFormat	
ErrorComms	
ErrorAccessDenied	
ErrorNoData	
ErrorEndOfData	
ErrorDataPresent	
ErrorDataTruncated	
ErrorApiVersion	
ErrorAppBase	
ErrorUserBase	

8.36 BErrorTime.cpp File Reference

```
#include <BErrorTime.h>
```

8.37 BErrorTime.h File Reference

```
#include <BString.h>
#include <BTimeStamp.h>
```

Classes

- class BErrorTime

  Error return class with time field.

8.38 BEvent.cpp File Reference

```
#include <BEvent.h>
#include <BPoll.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
```

8.39 BEvent.h File Reference

```
#include <BTypes.h>
#include <BQueue.h>
```

Classes

- class BEvent

  An event description class.
- class BEventPipe

  This class provides an interface for sending simple integer events via a pipe file descriptor.

Typedefs

- typedef BQueue< BEvent > BEventQueue

  This class provides an interface for sending simple integer events via a BQueue.

8.39.1 Typedef Documentation

8.39.1.1 BEventQueue

```
typedef BQueue<BEvent> BEventQueue
```

This class provides an interface for sending simple integer events via a BQueue.

8.40 BEvent1.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <BEvent1.h>
#include <BPoll.h>
```

8.41 BEvent1.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BEvent1

  This class provides a base class for all event objects that can be sent over the events interface.
- class BEvent1Error

  This class provides a class to send a BError event.
- class BEvent1Pipe

  This class provides a base interface for sending events via a pipe. This allows threads to send events that can be picked up by the poll system call.
- class BEvent1Int

  This class provides an interface for sending simple integer events via a file descriptor. This allows threads to send events that can be picked up by the poll system call.

Enumerations

- enum BEvent1Type { BEvent1TypeNone , BEvent1TypeInt , BEvent1TypeError }

8.41.1 Enumeration Type Documentation

8.41.1.1 BEvent1Type

```
enum BEvent1Type
```

Enumerator

BEvent1TypeNone	
BEvent1TypeInt	
BEvent1TypeError	

8.42 BFifo.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BFifo.inc>
```

Classes

- class BFifo< Type >

  A template first in first out data buffer to store any object types.

8.43 BFifo.inc File Reference

8.44 BFifoCirc.cpp File Reference

```
#include <BFifoCirc.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/mman.h>
```

Macros

- #define dprintf(fmt, a...)

8.44.1 Macro Definition Documentation

8.44.1.1 dprintf

```
#define dprintf(
             fmt,
             a...  )
```

8.45 BFifoCirc.h File Reference

```
#include <stdint.h>
#include <BError.h>
#include <BCondInt.h>
#include <BMutex.h>
#include <BFifoCirc.inc>
```

Classes

- class BFifoCircPos

  This class implements a pointer into the Fifo's circular buffer.
- class BFifoCirc< Type >

  This class implements a thread safe FIFO buffer using a binary sized circular memory.

8.46 BFifoCirc.inc File Reference

8.47 BFile.cpp File Reference

```
#include <stdarg.h>
#include <BFile.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
```

Macros

- #define STRBUF 10240

8.47.1 Macro Definition Documentation

8.47.1.1 STRBUF

```
#define STRBUF 10240
```

8.48 BFile.h File Reference

```
#include <stdio.h>
#include <BTypes.h>
#include <BString.h>
#include <BError.h>
```

Classes

- class BFile

  File operations class.

8.49 BFileCsv.cpp File Reference

```
#include <BFileCsv.h>
#include <errno.h>
```

8.50 BFileCsv.h File Reference

```
#include <BFile.h>
```

Classes

- class BFileCsv

  A class to read and write CSV formatted files.

8.51 BFileData.cpp File Reference

```
#include <BFileCsv.h>
#include <BFileData.h>
#include <errno.h>
```

8.52 BFileData.h File Reference

```
#include <BError.h>
```

Classes

- class BFileData

  A class to implement a data storage file.

8.53 BFirmware.h File Reference

```
#include <BTypes.h>
```

Classes

- struct BFirmwareFileHeader
- struct BFirmwareSegHeader
- struct BFirmwareInfo

Typedefs

- typedef BFirmwareFileHeader BFirmwareFirmwareHeader

Functions

- struct BFirmwareFileHeader __attribute__ ((packed))
- int bfirmwareValid (BUInt32 baseAddress, BUInt type, Bool checkChecksum, char *version=0)
- void bfirmwareBoot (BUInt32 baseAddress)

Variables

- const BUInt32 BFirmwareMagic = 0x01414542
- const BUInt32 BFirmwareTypeFile = 1
- const BUInt32 BFirmwareTypeFirmware = 2
- const BUInt32 BFirmwareTypeSegment = 3
- const BUInt32 BFirmwareFormatRaw = 0
- const BUInt32 BFirmwareFormatGzip = 1
- const BUInt32 BFirmwarePlatformBMeasure125 = 33
- const BUInt32 BFirmwarePlatformBMeasure125Cpu = 34
- const BUInt32 BFirmwarePlatformBMeasure125Fpga = 35
- const BUInt32 BFirmwarePlatformBMeasure125Wifi = 36
- const BUInt32 BFirmwarePlatformBMeasure125Boot = 37
- BUInt32 magic
- BUInt32 itemType
- BUInt32 fileLength
- BUInt32 checksum
- BUInt32 platform
- BUInt32 format
- BUInt32 numSegments
- BUInt32 startAddress
- BUInt8 ver0
- BUInt8 ver1
- BUInt8 ver2
- BUInt8 ver3
- BUInt32 special [7]
- BUInt32 dataLength
- BUInt32 address
- BUInt32 length
- const BUInt32 BFirmwareInfoMagic = 0xBBEEAA00
- const BUInt8 BFirmwareInfoEncrypt1 = 0x40
- struct BFirmwareInfo __attribute__

8.53.1 Typedef Documentation

8.53.1.1 BFirmwareFirmwareHeader

```
typedef BFirmwareFileHeader BFirmwareFirmwareHeader
```

8.53.2 Function Documentation

8.53.2.1 __attribute__()

```
struct BFirmwareFileHeader __attribute__ (
            (packed)  )
```

8.53.2.2 bfirmwareValid()

```
int bfirmwareValid (
            BUInt32 baseAddress,
            BUInt type,
            Bool checkChecksum,
            char * version = 0 )
```

8.53.2.3 bfirmwareBoot()

```
void bfirmwareBoot (
            BUInt32 baseAddress )
```

8.53.3 Variable Documentation

8.53.3.1 BFirmwareMagic

```
const BUInt32 BFirmwareMagic = 0x01414542
```

8.53.3.2 BFirmwareTypeFile

```
const BUInt32 BFirmwareTypeFile = 1
```

8.53.3.3 BFirmwareTypeFirmware

```
const BUInt32 BFirmwareTypeFirmware = 2
```

8.53.3.4 BFirmwareTypeSegment

```
const BUInt32 BFirmwareTypeSegment = 3
```

8.53.3.5 BFirmwareFormatRaw

```
const BUInt32 BFirmwareFormatRaw = 0
```

8.53.3.6 BFirmwareFormatGzip

```
const BUInt32 BFirmwareFormatGzip = 1
```

8.53.3.7 BFirmwarePlatformBMeasure125

```
const BUInt32 BFirmwarePlatformBMeasure125 = 33
```

8.53.3.8 BFirmwarePlatformBMeasure125Cpu

```
const BUInt32 BFirmwarePlatformBMeasure125Cpu = 34
```

8.53.3.9 BFirmwarePlatformBMeasure125Fpga

```
const BUInt32 BFirmwarePlatformBMeasure125Fpga = 35
```

8.53.3.10 BFirmwarePlatformBMeasure125Wifi

```
const BUInt32 BFirmwarePlatformBMeasure125Wifi = 36
```

8.53.3.11 BFirmwarePlatformBMeasure125Boot

```
const BUInt32 BFirmwarePlatformBMeasure125Boot = 37
```

8.53.3.12 magic

```
BUInt32 magic
```

8.53.3.13 itemType

```
BUInt32 itemType
```

8.53.3.14 fileLength

```
BUInt32 fileLength
```

8.53.3.15 checksum

```
BUInt32 checksum
```

8.53.3.16 platform

```
BUInt32 platform
```

8.53.3.17 format

```
BUInt32 format
```

8.53.3.18 numSegments

```
BUInt32 numSegments
```

8.53.3.19 startAddress

```
BUInt32 startAddress
```

8.53.3.20 ver0

```
BUInt8 ver0
```

8.53.3.21 ver1

```
BUInt8 ver1
```

8.53.3.22 ver2

```
BUInt8 ver2
```

8.53.3.23 ver3

```
BUInt8 ver3
```

8.53.3.24 special

```
BUInt32 special
```

8.53.3.25 dataLength

```
BUInt32 dataLength
```

8.53.3.26 address

```
BUInt32 address
```

8.53.3.27 length

```
BUInt32 length
```

8.53.3.28 BFirmwareInfoMagic

```
const BUInt32 BFirmwareInfoMagic = 0xBBEEAA00
```

8.53.3.29 BFirmwareInfoEncrypt1

```
const BUInt8 BFirmwareInfoEncrypt1 = 0x40
```

8.53.3.30 __attribute__

```
struct BoapMc1Error __attribute__
```

8.54 BList.h File Reference

```
#include <BList_func.h>
```

Classes

- class BNode
  A BList entry's node.
- class BIter
  Iterator for BLists.
- class BList< T >
  Template based list class.
- class BList< T >::Node
  A BList internal Node.

Macros

- #define BListLoop(list, i) for(BIter i = list.begin(); !list.isEnd(i); list.next(i))

8.54.1 Macro Definition Documentation

8.54.1.1 BListLoop

```
#define BListLoop(
             list,
             i ) for(BIter i = list.begin(); !list.isEnd(i); list.next(i))
```

8.55 BList_func.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <memory.h>
```

8.56 BMutex.cpp File Reference

```
#include <BMutex.h>
```

Macros

- #define MDEBUG 0

8.56.1 Macro Definition Documentation

8.56.1.1 MDEBUG

```
#define MDEBUG 0
```

8.57 BMutex.h File Reference

```
#include <pthread.h>
```

Classes

- class BMutex

  Mutex class. Note these are recursive Mutexes and so you need to make sure the number of unlocks equals the number of locks.
- class BMutexLock

  Mutex class that removes the lock on deletion and so is useful to lock data in a function call.

8.58 BMysql.cpp File Reference

```
#include <stdlib.h>
#include <string.h>
#include <BMysql.h>
```

8.59 BMysql.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BDict.h>
#include <BMutex.h>
#include <mysql/mysql.h>
```

Classes

- class BMysql

  A class to provide access to a MySQL database.

8.60 BNameValue.h File Reference

```
#include <BList.h>
#include <BString.h>
```

Classes

- class BNameValue< T >

  A simple, templated, name/value pair.
- class BNameValueList< T >

  A simple, templated, name/value pair list.

8.61 Boap.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <Boap.h>
#include <byteswap.h>
#include <BoapnsD.h>
#include <BoapnsC.h>
```

Macros

- #define DEBUG 0
- #define APIVERSION_TEST 1
- #define dprintf(fmt, a...)
- #define IS_BIG_ENDIAN 1

Variables

- const int boapPort = 12000

  The default BOAP connection port.

8.61.1 Macro Definition Documentation

8.61.1.1 DEBUG

```
#define DEBUG 0
```

8.61.1.2 APIVERSION_TEST

```
#define APIVERSION_TEST 1
```

8.61.1.3 dprintf

```
#define dprintf(
              fmt,
              a...  )
```

8.61.1.4 IS_BIG_ENDIAN

```
#define IS_BIG_ENDIAN 1
```

8.61.2 Variable Documentation

8.61.2.1 boapPort

```
const int boapPort = 12000
```

The default BOAP connection port.

8.62 Boap.h File Reference

```
#include <stdint.h>
#include <BTypes.h>
#include <BPoll.h>
#include <BSocket.h>
#include <BThread.h>
#include <BError.h>
#include <BEvent1.h>
#include <BMutex.h>
#include <BTimeStamp.h>
#include <BBuffer.h>
```

Classes

- struct BoapPacketHead

  Boap packet header.
- class BoapPacket

  Boap packet.
- class BoapClientObject

  Base for all Boap client objects.
- class BoapSignalObject

  A Boap object to send signals using an RPC mechanism.
- class BoapServiceEntry

  Boap server single service entry.
- class BoapServerConnection

  Boap server connection.
- class BoapServer

  Boap server.
- class BoapFuncEntry

  Boap service function.
- class BoapServiceObject

  Boap service object.

Namespaces

- Boapns

Typedefs

- typedef BUInt32 BoapService
- typedef BError(BoapServiceObject::* BoapFunc) (BoapServerConnection *conn, BoapPacket &rx, BoapPacket &tx)

Enumerations

- enum BoapType { BoapTypeRpc , BoapTypeRpcReply , BoapTypeSignal , BoapTypeRpcError , BoapTypeRpc , BoapTypeSignal }
- enum BoapPriority { BoapPriorityLow , BoapPriorityNormal , BoapPriorityHigh }

Variables

- const BUInt32 BoapMagic = 0x424F4100

8.62.1 Typedef Documentation

8.62.1.1 BoapService

```
typedef BUInt32 BoapService
```

8.62.1.2 BoapFunc

```
typedef BError(BoapServiceObject::* BoapFunc) (BoapServerConnection *conn, BoapPacket &rx,
BoapPacket &tx)
```

8.62.2 Enumeration Type Documentation

8.62.2.1 BoapType

```
enum BoapType
```

Enumerator

BoapTypeRpc	
BoapTypeRpcReply	
BoapTypeSignal	
BoapTypeRpcError	
BoapTypeRpc	
BoapTypeSignal	

8.62.2.2 BoapPriority

```
enum BoapPriority
```

Enumerator

BoapPriorityLow	
BoapPriorityNormal	
BoapPriorityHigh	

8.62.3 Variable Documentation

8.62.3.1 BoapMagic

```
const BUInt32 BoapMagic = 0x424F4100
```

8.63 BoapMc.cpp File Reference

```
#include <BoapMc.h>
#include <BCrc16.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

Macros

- #define DEBUG_LOCAL 0
- #define DEBUG_LOCAL1 0
- #define dlprintf(fmt, a...)
- #define dl1printf(fmt, a...)

8.63.1 Macro Definition Documentation

8.63.1.1 DEBUG_LOCAL

```
#define DEBUG_LOCAL 0
```

8.63.1.2 DEBUG_LOCAL1

```
#define DEBUG_LOCAL1 0
```

8.63.1.3 dlprintf

```
#define dlprintf(
              fmt,
              a...  )
```

8.63.1.4 dl1printf

```
#define dl1printf(
              fmt,
              a...  )
```

8.64 BoapMc.h File Reference

```
#include <BTypes.h>
#include <BMutex.h>
#include <BSemaphore.h>
#include <BQueue.h>
#include <BFifo.h>
#include <BComms.h>
```

Classes

- struct BoapMcPacketHead
- class BoapMcPacket
- class BoapMcClientObject
- class BoapMcSignalObject
- class BoapMcServiceObject
- class BoapMcComms

Enumerations

- enum BoapMcType { BoapMcTypeRequest = 0x00 , BoapMcTypeReply = 0x80 }

Functions

- struct BoapMcPacketHead __attribute__ ((aligned(8), packed))

Variables

- BUInt8 length
- BUInt8 addressTo
- BUInt8 addressFrom
- BUInt8 cmd
- BUInt16 error
- BUInt16 checksum
- class BoapMcPacket __attribute__

8.64.1 Enumeration Type Documentation

8.64.1.1 BoapMcType

```
enum BoapMcType
```

Enumerator

BoapMcTypeRequest	
BoapMcTypeReply	

8.64.2 Function Documentation

8.64.2.1 __attribute__()

```
struct BoapMcPacketHead __attribute__ (
            (aligned(8), packed)  )
```

8.64.3 Variable Documentation

8.64.3.1 length

```
BUInt8 length
```

8.64.3.2 addressTo

```
BUInt8 addressTo
```

8.64.3.3 addressFrom

```
BUInt8 addressFrom
```

8.64.3.4 cmd

```
BUInt8 cmd
```

8.64.3.5 error

```
BUInt16 error
```

8.64.3.6 checksum

```
BUInt16 checksum
```

8.64.3.7 __attribute__

```
class BoapMcPacket __attribute__
```

8.65 BoapMc1.cpp File Reference

```
#include <BoapMc1.h>
#include <BSys.h>
#include <BCrc32.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <BDebug.h>
```

Macros

- #define BDEBUGL1 0
- #define BDEBUGL2 0

8.65.1 Macro Definition Documentation

8.65.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.65.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.66 BoapMc1.h File Reference

```
#include <BTypes.h>
#include <BMutex.h>
#include <BSemaphore.h>
#include <BQueue.h>
#include <BFifo.h>
#include <BComms.h>
```

Classes

- struct BoapMc1PacketHead
- class BoapMc1Packet
- struct BoapMc1Error
- class BoapMc1Comms

Enumerations

- enum BoapMc1Type { BoapMc1TypeRequest = 0x0000 , BoapMc1TypeReply = 0x8000 }

Functions

- struct BoapMc1PacketHead __attribute__ ((aligned(8), packed))
- BUInt32 boapMc1CommsRoundupLen (BUInt32 len)

Variables

- const BUInt16 BoapMc1Magic = 0x5542
- BUInt16 magic

  Packet magic pattern.
- BUInt16 length

  Total packet length including the header.
- BUInt16 addressTo

  Address to send to.
- BUInt16 addressFrom

  Address packet is from.
- BUInt16 cmd

  The RPC command or reply number.
- BInt16 error

  Error number.
- BUInt32 checksum

  Packet checksum, when used.
- BoapMc1PacketHead head
- char data [8]
- BInt16 number

  The error number.
- char string [32]

  The error string.
- class BoapMc1Comms __attribute__

8.66.1 Enumeration Type Documentation

8.66.1.1 BoapMc1Type

```
enum BoapMc1Type
```

Enumerator

BoapMc1TypeRequest	
BoapMc1TypeReply	

8.66.2 Function Documentation

8.66.2.1 __attribute__()

```
struct BoapMc1PacketHead __attribute__ (
            (aligned(8), packed)  )
```

8.66.2.2 boapMc1CommsRoundupLen()

```
BUInt32 boapMc1CommsRoundupLen (
            BUInt32 len )  [inline]
```

8.66.3 Variable Documentation

8.66.3.1 BoapMc1Magic

```
const BUInt16 BoapMc1Magic = 0x5542
```

8.66.3.2 magic

```
BUInt16 magic
```

Packet magic pattern.

8.66.3.3 length

```
BUInt16 length
```

Total packet length including the header.

8.66.3.4 addressTo

```
BUInt16 addressTo
```

Address to send to.

8.66.3.5 addressFrom

```
BUInt16 addressFrom
```

Address packet is from.

8.66.3.6 cmd

```
BUInt16 cmd
```

The RPC command or reply number.

8.66.3.7 error

```
BInt16 error
```

Error number.

8.66.3.8 checksum

```
BUInt32 checksum
```

Packet checksum, when used.

8.66.3.9 head

```
BoapMc1PacketHead head
```

8.66.3.10 data

```
char data[8]
```

8.66.3.11 number

```
BInt16 number
```

The error number.

8.66.3.12 string

```
char string[32]
```

The error string.

8.66.3.13 __attribute__

```
class BoapMc1Comms __attribute__
```

8.67 BoapnsC.cpp File Reference

```
#include <BoapnsC.h>
```

Namespaces

- Boapns

8.68 BoapnsC.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <Boap.h>
#include <BString.h>
#include <BList.h>
#include <BArray.h>
#include <BoapnsD.h>
```

Classes

- class Boapns::Boapns

Namespaces

- Boapns

Variables

- const BUInt32 Boapns::apiVersion = 0

8.69 BoapnsD.cpp File Reference

```
#include <BoapnsD.h>
```

Namespaces

- Boapns

8.70 BoapnsD.h File Reference

BOAP data class definitions for: Boapns.

```
#include <Boap.h>
#include <BObj.h>
#include <BDate.h>
#include <BTimeStamp.h>
#include <BComplex.h>
#include <BList.h>
#include <BArray.h>
```

Classes

- class Boapns::BoapEntry

Namespaces

- Boapns

8.70.1 Detailed Description

BOAP data class definitions for: Boapns.

Date

2022-11-30T14:15:59

The classes in here have been defined by a BOAP ∗.bidl file and define classes able to be communicated across a BOAP link

8.71 BoapSimple.cc File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <Boap.h>
#include <BoapnsD.h>
#include <BoapnsC.h>
```

Macros

- #define DEBUG 0
- #define dprintf(fmt, a...)

Variables

- const int roundSize = 256

8.71.1 Macro Definition Documentation

8.71.1.1 DEBUG

```
#define DEBUG 0
```

8.71.1.2 dprintf

```
#define dprintf(
              fmt,
              a...  )
```

8.71.2 Variable Documentation

8.71.2.1 roundSize

```
const int roundSize = 256
```

8.72 BoapSimple.h File Reference

```
#include <stdint.h>
#include <BPoll.h>
#include <BSocket.h>
#include <BError.h>
```

Classes

- struct BoapPacketHead

  Boap packet header.
- class BoapPacket

  Boap packet.
- class BoapClientObject

  Base for all Boap client objects.
- class BoapSignalObject

  A Boap object to send signals using an RPC mechanism.
- class BoapServiceEntry

  Boap server single service entry.
- class BoapServer

  Boap server.
- class BoapFuncEntry

  Boap service function.
- class BoapServiceObject

  Boap service object.

Typedefs

- typedef int8_t Int8
- typedef uint8_t UInt8
- typedef int16_t Int16
- typedef uint16_t UInt16
- typedef int32_t Int32
- typedef uint32_t UInt32
- typedef double Double
- typedef uint32_t BoapService
- typedef BError(BoapServiceObject::* BoapFunc) (BoapPacket &rx, BoapPacket &tx)

Enumerations

- enum BoapType {
  BoapTypeRpc , BoapTypeRpcReply , BoapTypeSignal , BoapTypeRpcError ,
  BoapTypeRpc , BoapTypeSignal }

8.72.1 Typedef Documentation

8.72.1.1 Int8

```
typedef int8_t Int8
```

8.72.1.2 UInt8

```
typedef uint8_t UInt8
```

8.72.1.3 Int16

```
typedef int16_t Int16
```

8.72.1.4 UInt16

```
typedef uint16_t UInt16
```

8.72.1.5 Int32

```
typedef int32_t Int32
```

8.72.1.6 UInt32

```
typedef uint32_t UInt32
```

8.72.1.7 Double

```
typedef double Double
```

8.72.1.8 BoapService

```
typedef uint32_t BoapService
```

8.72.1.9 BoapFunc

```
typedef BError(BoapServiceObject::* BoapFunc) (BoapPacket &rx, BoapPacket &tx)
```

8.72.2 Enumeration Type Documentation

8.72.2.1 BoapType

```
enum BoapType
```

Enumerator

BoapTypeRpc	
BoapTypeRpcReply	
BoapTypeSignal	
BoapTypeRpcError	
BoapTypeRpc	
BoapTypeSignal	

8.73 BObj.cpp File Reference

```
#include <BObj.h>
```

8.74 BObj.h File Reference

```
#include <BTypes.h>
#include <BDict.h>
```

```
#include <BString.h>
#include <BError.h>
```

Classes

- class BObj

  A generic object base class that has runtime definable data feilds.

8.75 BObjStringFormat.cpp File Reference

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

Functions

- BString toBString (BString n, Bool v)

  A set of functions to perform object to string and string to object for standard types and generic BObj classes.
- BString toBString (BString n, BInt8 v)
- BString toBString (BString n, BUInt8 v)
- BString toBString (BString n, BInt16 v)
- BString toBString (BString n, BUInt16 v)
- BString toBString (BString n, BInt32 v)
- BString toBString (BString n, BUInt32 v)
- BString toBString (BString n, BInt64 v)
- BString toBString (BString n, BUInt64 v)
- BString toBString (BString n, BFloat32 v)
- BString toBString (BString n, BFloat64 v)
- BString toBString (BString n, BChar v)
- BString toBString (BString n, const BChar *v)
- BString toBString (BString n, BString v)
- BString toBString (BString n, BError v)
- BString toBString (BString n, BTime v)
- BString toBString (BString name, const BObjMember *m, const void *obj, BStringList ignoreFields)
- BString toBString (BString n, BObj &obj)
- BString toBStringJson (BString n, Bool v)
- BString toBStringJson (BString n, BInt8 v)
- BString toBStringJson (BString n, BUInt8 v)
- BString toBStringJson (BString n, BInt16 v)
- BString toBStringJson (BString n, BUInt16 v)
- BString toBStringJson (BString n, BInt32 v)
- BString toBStringJson (BString n, BUInt32 v)
- BString toBStringJson (BString n, BInt64 v)
- BString toBStringJson (BString n, BUInt64 v)
- BString toBStringJson (BString n, BFloat32 v)
- BString toBStringJson (BString n, BFloat64 v)
- BString toBStringJson (BString n, BChar v)
- BString toBStringJson (BString n, const BChar *v)
- BString toBStringJson (BString n, BString v)
- BString toBStringJson (BString n, BError v)
- BString toBStringJson (BString n, BTime v)
- BString toBStringJson (BString n, const BObjMember *m, const void *obj, BStringList ignoreFields)
- BString toBStringJson (BString n, BObj &obj)
- BError toBDictStringFromJson (BString json, BDictString &ds)

8.75.1 Function Documentation

8.75.1.1 toBString() [1/18]

```
BString toBString (
            BString n,
            Bool v )
```

A set of functions to perform object to string and string to object for standard types and generic BObj classes.

8.75.1.2 toBString() [2/18]

```
BString toBString (
            BString n,
            BInt8 v )
```

8.75.1.3 toBString() [3/18]

```
BString toBString (
            BString n,
            BUInt8 v )
```

8.75.1.4 toBString() [4/18]

```
BString toBString (
            BString n,
            BInt16 v )
```

8.75.1.5 toBString() [5/18]

```
BString toBString (
            BString n,
            BUInt16 v )
```

8.75.1.6 toBString() [6/18]

```
BString toBString (
            BString n,
            BInt32 v )
```

8.75.1.7 toBString() [7/18]

```
BString toBString (
            BString n,
            BUInt32 v )
```

8.75.1.8 toBString() [8/18]

```
BString toBString (
            BString n,
            BInt64 v )
```

8.75.1.9 toBString() [9/18]

```
BString toBString (
            BString n,
            BUInt64 v )
```

8.75.1.10 toBString() [10/18]

```
BString toBString (
            BString n,
            BFloat32 v )
```

8.75.1.11 toBString() [11/18]

```
BString toBString (
            BString n,
            BFloat64 v )
```

8.75.1.12 toBString() [12/18]

```
BString toBString (
            BString n,
            BChar v )
```

8.75.1.13 toBString() [13/18]

```
BString toBString (
            BString n,
            const BChar * v )
```

8.75.1.14 toBString() [14/18]

```
BString toBString (
            BString n,
            BString v )
```

8.75.1.15 toBString() [15/18]

```
BString toBString (
            BString n,
            BError v )
```

8.75.1.16 toBString() [16/18]

```
BString toBString (
            BString n,
            BTime v )
```

8.75.1.17 toBString() [17/18]

```
BString toBString (
            BString name,
            const BObjMember * m,
            const void * obj,
            BStringList ignoreFields )
```

8.75.1.18 toBString() [18/18]

```
BString toBString (
            BString n,
            BObj & obj )
```

8.75.1.19 toBStringJson() [1/18]

```
BString toBStringJson (
            BString n,
            Bool v )
```

8.75.1.20 toBStringJson() [2/18]

```
BString toBStringJson (
            BString n,
            BInt8 v )
```

8.75.1.21 toBStringJson() [3/18]

```
BString toBStringJson (
            BString n,
            BUInt8 v )
```

8.75.1.22 toBStringJson() [4/18]

```
BString toBStringJson (
            BString n,
            BInt16 v )
```

8.75.1.23 toBStringJson() [5/18]

```
BString toBStringJson (
            BString n,
            BUInt16 v )
```

8.75.1.24 toBStringJson() [6/18]

```
BString toBStringJson (
            BString n,
            BInt32 v )
```

8.75.1.25 toBStringJson() [7/18]

```
BString toBStringJson (
            BString n,
            BUInt32 v )
```

8.75.1.26 toBStringJson() [8/18]

```
BString toBStringJson (
            BString n,
            BInt64 v )
```

8.75.1.27 toBStringJson() [9/18]

```
BString toBStringJson (
            BString n,
            BUInt64 v )
```

8.75.1.28 toBStringJson() [10/18]

```
BString toBStringJson (
            BString n,
            BFloat32 v )
```

8.75.1.29 toBStringJson() [11/18]

```
BString toBStringJson (
            BString n,
            BFloat64 v )
```

8.75.1.30 toBStringJson() [12/18]

```
BString toBStringJson (
            BString n,
            BChar v )
```

8.75.1.31 toBStringJson() [13/18]

```
BString toBStringJson (
            BString n,
            const BChar * v )
```

8.75.1.32 toBStringJson() [14/18]

```
BString toBStringJson (
            BString n,
            BString v )
```

8.75.1.33 toBStringJson() [15/18]

```
BString toBStringJson (
            BString n,
            BError v )
```

8.75.1.34 toBStringJson() [16/18]

```
BString toBStringJson (
            BString n,
            BTime v )
```

8.75.1.35 toBStringJson() [17/18]

```
BString toBStringJson (
            BString n,
            const BObjMember * m,
            const void * obj,
            BStringList ignoreFields )
```

8.75.1.36 toBStringJson() [18/18]

```
BString toBStringJson (
            BString n,
            BObj & obj )
```

8.75.1.37 toBDictStringFromJson()

```
BError toBDictStringFromJson (
            BString json,
            BDictString & ds )
```

8.76 BObjStringFormat.h File Reference

```
#include <BObj.h>
#include <BString.h>
#include <BTime.h>
```

Functions

- BString toBString (BString name, Bool value)

  A set of functions to perform object to string and string to object for standard types and generic BObj classes.
- BString toBString (BString name, BInt8 value)
- BString toBString (BString name, BUInt8 value)
- BString toBString (BString name, BInt16 value)
- BString toBString (BString name, BUInt16 value)
- BString toBString (BString name, BInt32 value)
- BString toBString (BString name, BUInt32 value)
- BString toBString (BString name, BInt64 value)
- BString toBString (BString name, BUInt64 value)
- BString toBString (BString name, BFloat32 value)
- BString toBString (BString name, BFloat64 value)
- BString toBString (BString name, BChar value)
- BString toBString (BString name, const BChar *value)
- BString toBString (BString name, BString value)
- BString toBString (BString name, BError value)
- BString toBString (BString name, BTime time)
- BString toBString (BString name, const BObjMember *members, const void *obj, BStringList ignore↩ Fields=BStringList())
- BString toBString (BString name, BObj &obj)
- BString toBStringJson (BString name, Bool value)
- BString toBStringJson (BString name, BInt8 value)
- BString toBStringJson (BString name, BUInt8 value)
- BString toBStringJson (BString name, BInt16 value)
- BString toBStringJson (BString name, BUInt16 value)
- BString toBStringJson (BString name, BInt32 value)
- BString toBStringJson (BString name, BUInt32 value)

- BString toBStringJson (BString name, BInt64 value)
- BString toBStringJson (BString name, BUInt64 value)
- BString toBStringJson (BString name, BFloat32 value)
- BString toBStringJson (BString name, BFloat64 value)
- BString toBStringJson (BString name, BChar value)
- BString toBStringJson (BString name, const BChar *value)
- BString toBStringJson (BString name, BString value)
- BString toBStringJson (BString name, BError value)
- BString toBStringJson (BString name, BTime time)
- BString toBStringJson (BString name, const BObjMember *members, const void *obj, BStringList ignore↩ Fields=BStringList())
- BString toBStringJson (BString name, BObj &obj)
- BError toBDictStringFromJson (BString json, BDictString &ds)
- BString base64_encode (void *data, BUInt len)
- BError base64_decode (BString strIn, BString &strOut)

8.76.1 Function Documentation

8.76.1.1 toBString() [1/18]

```
BString toBString (
            BString name,
            Bool value )
```

A set of functions to perform object to string and string to object for standard types and generic BObj classes.

8.76.1.2 toBString() [2/18]

```
BString toBString (
            BString name,
            BInt8 value )
```

8.76.1.3 toBString() [3/18]

```
BString toBString (
            BString name,
            BUInt8 value )
```

8.76.1.4 toBString() [4/18]

```
BString toBString (
            BString name,
            BInt16 value )
```

8.76.1.5 toBString() [5/18]

```
BString toBString (
            BString name,
            BUInt16 value )
```

8.76.1.6 toBString() [6/18]

```
BString toBString (
            BString name,
            BInt32 value )
```

8.76.1.7 toBString() [7/18]

```
BString toBString (
            BString name,
            BUInt32 value )
```

8.76.1.8 toBString() [8/18]

```
BString toBString (
            BString name,
            BInt64 value )
```

8.76.1.9 toBString() [9/18]

```
BString toBString (
            BString name,
            BUInt64 value )
```

8.76.1.10 toBString() [10/18]

```
BString toBString (
            BString name,
            BFloat32 value )
```

8.76.1.11 toBString() [11/18]

```
BString toBString (
            BString name,
            BFloat64 value )
```

8.76.1.12 toBString() [12/18]

```
BString toBString (
            BString name,
            BChar value )
```

8.76.1.13 toBString() [13/18]

```
BString toBString (
            BString name,
            const BChar * value )
```

8.76.1.14 toBString() [14/18]

```
BString toBString (
            BString name,
            BString value )
```

8.76.1.15 toBString() [15/18]

```
BString toBString (
            BString name,
            BError value )
```

8.76.1.16 toBString() [16/18]

```
BString toBString (
            BString name,
            BTime time )
```

8.76.1.17 toBString() [17/18]

```
BString toBString (
            BString name,
            const BObjMember * members,
            const void * obj,
            BStringList ignoreFields = BStringList() )
```

8.76.1.18 toBString() [18/18]

```
BString toBString (
            BString name,
            BObj & obj )
```

8.76.1.19 toBStringJson() [1/18]

```
BString toBStringJson (
            BString name,
            Bool value )
```

8.76.1.20 toBStringJson() [2/18]

```
BString toBStringJson (
            BString name,
            BInt8 value )
```

8.76.1.21 toBStringJson() [3/18]

```
BString toBStringJson (
            BString name,
            BUInt8 value )
```

8.76.1.22 toBStringJson() [4/18]

```
BString toBStringJson (
            BString name,
            BInt16 value )
```

8.76.1.23 toBStringJson() [5/18]

```
BString toBStringJson (
            BString name,
            BUInt16 value )
```

8.76.1.24 toBStringJson() [6/18]

```
BString toBStringJson (
            BString name,
            BInt32 value )
```

8.76.1.25 toBStringJson() [7/18]

```
BString toBStringJson (
            BString name,
            BUInt32 value )
```

8.76.1.26 toBStringJson() [8/18]

```
BString toBStringJson (
            BString name,
            BInt64 value )
```

8.76.1.27 toBStringJson() [9/18]

```
BString toBStringJson (
            BString name,
            BUInt64 value )
```

8.76.1.28 toBStringJson() [10/18]

```
BString toBStringJson (
            BString name,
            BFloat32 value )
```

8.76.1.29 toBStringJson() [11/18]

```
BString toBStringJson (
            BString name,
            BFloat64 value )
```

8.76.1.30 toBStringJson() [12/18]

```
BString toBStringJson (
            BString name,
            BChar value )
```

8.76.1.31 toBStringJson() [13/18]

```
BString toBStringJson (
            BString name,
            const BChar * value )
```

8.76.1.32 toBStringJson() [14/18]

```
BString toBStringJson (
            BString name,
            BString value )
```

8.76.1.33 toBStringJson() [15/18]

```
BString toBStringJson (
            BString name,
            BError value )
```

8.76.1.34 toBStringJson() [16/18]

```
BString toBStringJson (
            BString name,
            BTime time )
```

8.76.1.35 toBStringJson() [17/18]

```
BString toBStringJson (
            BString name,
            const BObjMember * members,
            const void * obj,
            BStringList ignoreFields = BStringList() )
```

8.76.1.36 toBStringJson() [18/18]

```
BString toBStringJson (
            BString name,
            BObj & obj )
```

8.76.1.37 toBDictStringFromJson()

```
BError toBDictStringFromJson (
            BString json,
            BDictString & ds )
```

8.76.1.38 base64_encode()

```
BString base64_encode (
            void * data,
            BUInt len )
```

8.76.1.39 base64_decode()

```
BError base64_decode (
            BString strIn,
            BString & strOut )
```

8.77 BPoll.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <BPoll.h>
```

8.78 BPoll.h File Reference

```
#include <BList.h>
#include <BError.h>
#include <sys/poll.h>
```

Classes

- class BPoll

  This class provides an interface for polling a number of file descriptors. It uses round robin polling.

8.79 BQueue.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BList.h>
#include <BMutex.h>
#include <BCondInt.h>
```

Classes

- class BQueue< T >

  Provides a thread save queue of objects that can be used to communicate between threads.

Typedefs

- typedef BQueue< BInt32 > BQueueInt

8.79.1 Typedef Documentation

8.79.1.1 BQueueInt

```
typedef BQueue<BInt32> BQueueInt
```

8.80 BRefData.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <BRefData.h>
```

Macros

- #define CHUNK 16

8.80.1 Macro Definition Documentation

8.80.1.1 CHUNK

```
#define CHUNK 16
```

8.81 BRefData.h File Reference

```
#include <BAtomicCount.h>
```

Classes

- class BRefData

  A pointer to a variable sized data area with reference counting so the data areas can be shared.

8.82 BRtc.cpp File Reference

```
#include <BRtc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/rtc.h>
```

8.83 BRtc.h File Reference

```
#include <BError.h>
#include <BThread.h>
#include <BCond.h>
```

Classes

- class BRtc

  Realtime clock for access to the systems real time battery backed up time hardware.
- class BRtcThreaded

  A thread safe class to access to the systems real time battery backed up time hardware.

8.84 BRWLock.cpp File Reference

```
#include <BRWLock.h>
```

8.85 BRWLock.h File Reference

```
#include <pthread.h>
```

Classes

- class BRWLock

  Thread read-write lock.

8.86 BSema.cpp File Reference

```
#include <BSema.h>
#include <errno.h>
#include <sys/time.h>
```

8.87 BSema.h File Reference

```
#include <sys/types.h>
#include <semaphore.h>
```

Classes

- class BSema

  Sempahore class.

8.88 BSemaphore.cpp File Reference

```
#include <BSemaphore.h>
#include <sys/time.h>
```

8.89 BSemaphore.h File Reference

```
#include <BTypes.h>
#include <BMutex.h>
#include <semaphore.h>
```

Classes

- class BSemaphore

  Base Semaphore class.
- class BSemaphoreBool

  Boolean semaphore.
- class BSemaphoreCount

  Integer counting semaphore.

8.90 BSocket.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <BSocket.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <net/if.h>
```

Macros

- #define IP_MTU 14

8.90.1 Macro Definition Documentation

8.90.1.1 IP_MTU

```
#define IP_MTU 14
```

8.91 BSocket.h File Reference

```
#include <BString.h>
#include <BError.h>
#include <BTypes.h>
#include <stdint.h>
#include <sys/types.h>
#include <netinet/in.h>
```

Classes

- class BSocketAddress

  Socket Address.
- class BSocketAddressINET

  IPV4 aware socket address.
- class BSocket

  A network communications socket.

Macros

- #define SOL_IP 0
- #define SO_PRIORITY 12
- #define MSG_NOSIGNAL 0

8.91.1 Macro Definition Documentation

8.91.1.1 SOL_IP

```
#define SOL_IP 0
```

8.91.1.2 SO_PRIORITY

```
#define SO_PRIORITY 12
```

8.91.1.3 MSG_NOSIGNAL

```
#define MSG_NOSIGNAL 0
```

8.92 BSpi.cpp File Reference

```
#include <BSpi.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <linux/spi/spidev.h>
```

8.93 BSpi.h File Reference

```
#include <BTypes.h>
#include <BError.h>
```

Classes

- class BSpi

  BSpi class for accessing SPI hardware devices.

8.94 BString.cpp File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <BString.h>
#include <BError.h>
#include <signal.h>
#include <regex.h>
```

Macros

- #define STRIP 0x7f
- #define MINUS '-'

Functions

- static int gmatch (const char *s, const char *p)
- std::ostream & operator<< (std::ostream &o, BString &s)
- std::istream & operator>> (std::istream &i, BString &s)
- int bstringListinList (BStringList &list, BString s)
- BString blistToString (const BStringList &list)

  Convert a string list to a comma separated string.
- BStringList bstringToList (BString str, int stripSpaces)

  Convert a comma separated string to a string list.
- BStringList charToList (const char **str)
- BString barrayToString (const BStringArray &list)

  Convert a string array to a comma separated string.
- BStringArray bstringToArray (BString str, int stripSpaces)

  Convert a comma separated string to a string array.
- BStringArray charToArray (const char **str)
- void toBString (BString &v, BString &s)
- void toBString (BStringList &v, BString &s)
- void toBString (BInt32 &v, BString &s)
- void toBString (BUInt32 &v, BString &s)
- void toBString (BUInt64 &v, BString &s)
- void toBString (BFloat64 &v, BString &s)
- void fromBString (BString &s, BString &v)
- void fromBString (BString &s, BStringList &v)
- void fromBString (BString &s, BInt32 &v)
- void fromBString (BString &s, BUInt32 &v)
- void fromBString (BString &s, BUInt64 &v)
- void fromBString (BString &s, BFloat64 &v)
- const char * intToString (char *str, BUInt strLen, int value, int base)
- const char * int64ToString (char *str, BUInt strLen, BInt64 value, int base)
- const char * floatToString (char *str, BUInt strLen, BFloat32 f, BUInt precision)
- char * bstrncpy (char *dest, const char *src, size_t n)
- char * bstrtrim (char *str)

Variables

- static const BUInt8 base64_decode_table [ ]

8.94.1 Macro Definition Documentation

8.94.1.1 STRIP

```
#define STRIP 0x7f
```

8.94.1.2 MINUS

```
#define MINUS '-'
```

8.94.2 Function Documentation

8.94.2.1 gmatch()

```
static int gmatch (
            const char * s,
            const char * p )  [static]
```

8.94.2.2 operator<<()

```
std::ostream& operator<< (
            std::ostream & o,
            BString & s )
```

8.94.2.3 operator>>()

```
std::istream& operator>> (
            std::istream & i,
            BString & s )
```

8.94.2.4 bstringListinList()

```
int bstringListinList (
            BStringList & list,
            BString s )
```

8.94.2.5 blistToString()

```
BString blistToString (
            const BStringList & list )
```

Convert a string list to a comma separated string.

8.94.2.6 bstringToList()

```
BStringList bstringToList (
            BString str,
            int stripSpaces )
```

Convert a comma separated string to a string list.

8.94.2.7 charToList()

```
BStringList charToList (
            const char ** str )
```

8.94.2.8 barrayToString()

```
BString barrayToString (
            const BStringArray & list )
```

Convert a string array to a comma separated string.

8.94.2.9 bstringToArray()

```
BStringArray bstringToArray (
            BString str,
            int stripSpaces )
```

Convert a comma separated string to a string array.

8.94.2.10 charToArray()

```
BStringArray charToArray (
            const char ** str )
```

8.94.2.11 toBString() [1/6]

```
void toBString (
            BString & v,
            BString & s )
```

8.94.2.12 toBString() [2/6]

```
void toBString (
            BStringList & v,
            BString & s )
```

8.94.2.13 toBString() [3/6]

```
void toBString (
            BInt32 & v,
            BString & s )
```

8.94.2.14 toBString() [4/6]

```
void toBString (
            BUInt32 & v,
            BString & s )
```

8.94.2.15 toBString() [5/6]

```
void toBString (
            BUInt64 & v,
            BString & s )
```

8.94.2.16 toBString() [6/6]

```
void toBString (
            BFloat64 & v,
            BString & s )
```

8.94.2.17 fromBString() [1/6]

```
void fromBString (
            BString & s,
            BString & v )
```

8.94.2.18 fromBString() [2/6]

```
void fromBString (
            BString & s,
            BStringList & v )
```

8.94.2.19 fromBString() [3/6]

```
void fromBString (
            BString & s,
            BInt32 & v )
```

8.94.2.20 fromBString() [4/6]

```
void fromBString (
            BString & s,
            BUInt32 & v )
```

8.94.2.21 fromBString() [5/6]

```
void fromBString (
            BString & s,
            BUInt64 & v )
```

8.94.2.22 fromBString() [6/6]

```
void fromBString (
            BString & s,
            BFloat64 & v )
```

8.94.2.23 intToString()

```
const char* intToString (
            char * str,
            BUInt strLen,
            int value,
            int base )
```

8.94.2.24 int64ToString()

```
const char* int64ToString (
            char * str,
            BUInt strLen,
            BInt64 value,
            int base )
```

8.94.2.25 floatToString()

```
const char* floatToString (
            char * str,
            BUInt strLen,
            BFloat32 f,
            BUInt precision )
```

8.94.2.26 bstrncpy()

```
char* bstrncpy (
            char * dest,
            const char * src,
            size_t n )
```

8.94.2.27 bstrtrim()

```
char* bstrtrim (
            char * str )
```

8.94.3 Variable Documentation

8.94.3.1 base64_decode_table

```
const BUInt8 base64_decode_table[]  [static]
```

Initial value:

```
= {
    66,66,66,66,66,66,66,66,66,66,64,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,62,66,66,66,63,52,53,
    54,55,56,57,58,59,60,61,66,66,66,65,66,66,66, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
    10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,66,66,66,66,66,66,26,27,28,
    29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,
    66,66,66,66,66,66
}
```

8.95 BString.h File Reference

```
#include <BTypes.h>
#include <BRefData.h>
#include <BList.h>
#include <BArray.h>
#include <iostream>
```

Classes

- class BString

  This class stores and manipulates ASCII strings.

Typedefs

- typedef BList< BString > BStringList
- typedef BArray< BString > BStringArray

Functions

- std::ostream & operator<< (std::ostream &o, BString &s)
- std::istream & operator>> (std::istream &i, BString &s)
- int bstringListinList (BStringList &l, BString s)
- BString blistToString (const BStringList &list)

  Convert a string list to a comma separated string.
- BStringList bstringToList (BString str, int stripSpaces=0)

  Convert a comma separated string to a string list.
- BStringList charToList (const char **str)
- BString barrayToString (const BStringArray &list)

  Convert a string array to a comma separated string.
- BStringArray bstringToArray (BString str, int stripSpaces=0)

  Convert a comma separated string to a string array.
- BStringArray charToArray (const char **str)
- void toBString (BString &v, BString &s)
- void toBString (BStringList &v, BString &s)
- void toBString (BInt32 &v, BString &s)
- void toBString (BUInt32 &v, BString &s)
- void toBString (BUInt64 &v, BString &s)
- void toBString (BFloat64 &v, BString &s)
- void fromBString (BString &s, BString &v)
- void fromBString (BString &s, BStringList &v)
- void fromBString (BString &s, BInt32 &v)
- void fromBString (BString &s, BUInt32 &v)
- void fromBString (BString &s, BUInt64 &v)
- void fromBString (BString &s, BFloat64 &v)
- char from_hex (char ch)
- char to_hex (char code)
- char * bstrncpy (char *dest, const char *src, size_t n)
- char * bstrtrim (char *str)
- const char * intToString (char *str, BUInt strLen, int value, int base=10)
- const char * int64ToString (char *str, BUInt strLen, BInt64 value, int base=10)
- const char * floatToString (char *str, BUInt strLen, BFloat32 f, BUInt precision)

8.95.1 Typedef Documentation

8.95.1.1 BStringList

```
typedef BList<BString> BStringList
```

8.95.1.2 BStringArray

```
typedef BArray<BString> BStringArray
```

8.95.2 Function Documentation

8.95.2.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & o,
            BString & s )
```

8.95.2.2 operator>>()

```
std::istream& operator>> (
            std::istream & i,
            BString & s )
```

8.95.2.3 bstringListinList()

```
int bstringListinList (
            BStringList & l,
            BString s )
```

8.95.2.4 blistToString()

```
BString blistToString (
            const BStringList & list )
```

Convert a string list to a comma separated string.

8.95.2.5 bstringToList()

```
BStringList bstringToList (
            BString str,
            int stripSpaces = 0 )
```

Convert a comma separated string to a string list.

8.95.2.6 charToList()

```
BStringList charToList (
            const char ** str )
```

8.95.2.7 barrayToString()

```
BString barrayToString (
            const BStringArray & list )
```

Convert a string array to a comma separated string.

8.95.2.8 bstringToArray()

```
BStringArray bstringToArray (
            BString str,
            int stripSpaces = 0 )
```

Convert a comma separated string to a string array.

8.95.2.9 charToArray()

```
BStringArray charToArray (
            const char ** str )
```

8.95.2.10 toBString() [1/6]

```
void toBString (
            BString & v,
            BString & s )
```

8.95.2.11 toBString() [2/6]

```
void toBString (
            BStringList & v,
            BString & s )
```

8.95.2.12 toBString() [3/6]

```
void toBString (
            BInt32 & v,
            BString & s )
```

8.95.2.13 toBString() [4/6]

```
void toBString (
            BUInt32 & v,
            BString & s )
```

8.95.2.14 toBString() [5/6]

```
void toBString (
            BUInt64 & v,
            BString & s )
```

8.95.2.15 toBString() [6/6]

```
void toBString (
            BFloat64 & v,
            BString & s )
```

8.95.2.16 fromBString() [1/6]

```
void fromBString (
            BString & s,
            BString & v )
```

8.95.2.17 fromBString() [2/6]

```
void fromBString (
            BString & s,
            BStringList & v )
```

8.95.2.18 fromBString() [3/6]

```
void fromBString (
            BString & s,
            BInt32 & v )
```

8.95.2.19 fromBString() [4/6]

```
void fromBString (
            BString & s,
            BUInt32 & v )
```

8.95.2.20 fromBString() [5/6]

```
void fromBString (
            BString & s,
            BUInt64 & v )
```

8.95.2.21 fromBString() [6/6]

```
void fromBString (
            BString & s,
            BFloat64 & v )
```

8.95.2.22 from_hex()

```
char from_hex (
            char ch )  [inline]
```

8.95.2.23 to_hex()

```
char to_hex (
            char code )  [inline]
```

8.95.2.24 bstrncpy()

```
char* bstrncpy (
            char * dest,
            const char * src,
            size_t n )
```

8.95.2.25 bstrtrim()

```
char* bstrtrim (
            char * str )
```

8.95.2.26 intToString()

```
const char* intToString (
            char * str,
            BUInt strLen,
            int value,
            int base = 10 )
```

8.95.2.27 int64ToString()

```
const char* int64ToString (
            char * str,
            BUInt strLen,
            BInt64 value,
            int base = 10 )
```

8.95.2.28 floatToString()

```
const char* floatToString (
            char * str,
            BUInt strLen,
            BFloat32 f,
            BUInt precision )
```

8.96 BStringLocked.h File Reference

```
#include <BString.h>
#include <BMutex.h>
```

Classes

- class BStringMutex

  Thread locked string internal mutex.
- class BStringLocked

  Provides a basic thread locked string.

8.97 BSys.cpp File Reference

```
#include <BSys.h>
#include <time.h>
```

Functions

- void delayUs (BUInt us)

  Will delay for given time in us, if tasks running task will sleep.
- void delayMs (BUInt ms)

  Will delay for given time in ms, if tasks running task will sleep.

8.97.1 Function Documentation

8.97.1.1 delayUs()

```
void delayUs (
            BUInt us )
```

Will delay for given time in us, if tasks running task will sleep.

8.97.1.2 delayMs()

```
void delayMs (
            BUInt ms )
```

Will delay for given time in ms, if tasks running task will sleep.

8.98 BSys.h File Reference

```
#include <BTypes.h>
```

Functions

- void delayUs (BUInt us)

  Will delay for given time in us, if tasks running task will sleep.
- void delayMs (BUInt ms)

  Will delay for given time in ms, if tasks running task will sleep.

8.98.1 Function Documentation

8.98.1.1 delayUs()

```
void delayUs (
            BUInt us )
```

Will delay for given time in us, if tasks running task will sleep.

8.98.1.2 delayMs()

```
void delayMs (
            BUInt ms )
```

Will delay for given time in ms, if tasks running task will sleep.

8.99 BTable.cpp File Reference

```
#include <BTable.h>
```

8.100 BTable.h File Reference

```
#include <BArray.h>
#include <BString.h>
```

Classes

- class BTable

  A simple string based table structure.

8.101 BTask.cpp File Reference

```
#include <BTask.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
```

8.102 BTask.h File Reference

```
#include <BError.h>
#include <pthread.h>
```

Classes

- class BTask

  Implements a thread of execution.

8.103 BThread.cpp File Reference

```
#include <BThread.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
```

8.104 BThread.h File Reference

```
#include <pthread.h>
```

Classes

- class BThread

  Implements a program execution thread.

8.105 BTime.cpp File Reference

```
#include <BTime.h>
```

Functions

- static bool yearIsLeap (BUInt16 year)
- static BUInt16 yearDays (BUInt16 year)

Variables

- static BUInt16 monDays [2][13]

8.105.1 Function Documentation

8.105.1.1 yearIsLeap()

```
static bool yearIsLeap (
            BUInt16 year )  [inline], [static]
```

8.105.1.2 yearDays()

```
static BUInt16 yearDays (
            BUInt16 year )  [inline], [static]
```

8.105.2 Variable Documentation

8.105.2.1 monDays

```
BUInt16 monDays[2][13]  [static]
```

Initial value:

```
= {
    { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
    { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.106 BTime.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BString.h>
```

Classes

- class BTime

  Implements a simple date/time class. Stores the date/time as a number of seconds since Unix epoch 1970-01-02T00:00:00.

8.107 BTimer.cpp File Reference

```
#include <BTimer.h>
#include <sys/time.h>
```

8.108 BTimer.h File Reference

```
#include <BMutex.h>
```

Classes

- class BTimer

  Stopwatch style timer.

8.109 BTimeStamp.cpp File Reference

```
#include <BTimeStamp.h>
#include <BTimeStampMs.h>
#include <math.h>
#include <sys/time.h>
```

Functions

- void toBString (BTimeStamp &v, BString &s)
- void fromBString (BString &s, BTimeStamp &v)

Variables

- static int mon_yday [2][13]

8.109.1 Function Documentation

8.109.1.1 toBString()

```
void toBString (
            BTimeStamp & v,
            BString & s )
```

8.109.1.2 fromBString()

```
void fromBString (
            BString & s,
            BTimeStamp & v )
```

8.109.2 Variable Documentation

8.109.2.1 mon_yday

```
int mon_yday[2][13]  [static]
```

Initial value:

```
= {
    { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
    { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.110 BTimeStamp.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BTimeStamp

  A date and time storage class with microsecond resolution.

Functions

- void toBString (BTimeStamp &v, BString &s)
- void fromBString (BString &s, BTimeStamp &v)

8.110.1 Function Documentation

8.110.1.1 toBString()

```
void toBString (
            BTimeStamp & v,
            BString & s )
```

8.110.1.2 fromBString()

```
void fromBString (
            BString & s,
            BTimeStamp & v )
```

8.111 BTimeStampMs.cpp File Reference

```
#include <BTimeStampMs.h>
#include <math.h>
#include <sys/time.h>
```

Variables

- static int mon_yday [2][13]

8.111.1 Variable Documentation

8.111.1.1 mon_yday

```
int mon_yday[2][13]  [static]
```

Initial value:

```
= {
    { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
    { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.112 BTimeStampMs.h File Reference

```
#include <stdint.h>
#include <BError.h>
```

Classes

- class BTimeStampMs

  A date and time storage class with milisecond resolution and an extra field to indicate a particular sampleNumber it reffers to.

8.113 BTimeUs.cpp File Reference

```
#include <BTimeUs.h>
#include <stdio.h>
```

Functions

- static bool yearIsLeap (BUInt16 year)
- static BUInt16 yearDays (BUInt16 year)

Variables

- static BUInt16 monDays [2][13]

8.113.1 Function Documentation

8.113.1.1 yearIsLeap()

```
static bool yearIsLeap (
            BUInt16 year )  [inline], [static]
```

8.113.1.2 yearDays()

```
static BUInt16 yearDays (
            BUInt16 year )  [inline], [static]
```

8.113.2 Variable Documentation

8.113.2.1 monDays

```
BUInt16 monDays[2][13]  [static]
```

Initial value:

```
= {
    { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365 },
    { 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335, 366 }
}
```

8.114 BTimeUs.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BString.h>
#include <BTime.h>
```

Classes

- class BTimeUs

  Time storage as an unsigned 64bit value to TAI standard.

8.115 BTypes.cpp File Reference

```
#include <BTypes.h>
```

Variables

- BUInt32 beamlibVersion = BeamlibVersion

8.115.1 Variable Documentation

8.115.1.1 beamlibVersion

```
BUInt32 beamlibVersion = BeamlibVersion
```

8.116 BTypes.h File Reference

```
#include <stdint.h>
#include <sys/types.h>
#include <vector>
```

Classes

- class BDataChunk

  A chunk of data allowing writes of multiple chunks of segmented data.
- struct BObjMember

  A structure to define a member of a generic BObj.

Macros

- #define BeamlibVersion 0x030000

Typedefs

- typedef bool Bool
- typedef int8_t BInt8
- typedef uint8_t BUInt8
- typedef int16_t BInt16
- typedef uint16_t BUInt16
- typedef int32_t BInt32
- typedef uint32_t BUInt32
- typedef int64_t BInt64
- typedef uint64_t BUInt64
- typedef float BFloat32
- typedef double BFloat64
- typedef char BChar
- typedef BInt32 BInt
- typedef BUInt32 BUInt
- typedef BFloat32 BFloat
- typedef BFloat64 BDouble
- typedef size_t BSize
- typedef std::vector< BFloat32 > BArrayFloat
- typedef std::vector< BFloat64 > BArrayDouble
- typedef BUInt32 BTimeout

Enumerations

- enum BEventType {
  BEventTypeNone , BEventTypeError , BEventTypeRead , BEventTypeReadLine ,
  BEventTypeWrite , BEventTypeConnect , BEventTypeDisconnect , BEventTypeClientConnect ,
  BEventTypeClientDisconnect }
- enum BEventWaitSet {
  BEventWaitNone = 0x00 , BEventWaitError = 0x01 , BEventWaitRead = 0x02 , BEventWaitReadLine = 0x04
  ,
  BEventWaitWrite = 0x08 , BEventWaitConnect = 0x10 , BEventWaitDisconnect = 0x20 , BEventWaitClientConnect
  = 0x40 ,
  BEventWaitClientDisconnect = 0x80 , BEventWaitAny = 0xFFFFFFFF }
- enum BType {
  BTypeNone , BTypeBool , BTypeInt8 , BTypeUInt8 ,
  BTypeInt16 , BTypeUInt16 , BTypeInt32 , BTypeUInt32 ,
  BTypeInt64 , BTypeUInt64 , BTypeFloat32 , BTypeFloat64 ,
  BTypeChar , BTypeString , BTypeError , BTypeTime ,
  BTypeTimeUs , BTypeObj = 100 }
- enum BTypeComp {
  BTypeCompSingle , BTypeCompArray , BTypeCompArrayFixed , BTypeCompList ,
  BTypeCompDict }

Functions

- BTimeout timeoutTicks (BTimeout timeoutUs)
- void byteSwap8 (void *d, void *s)
- void byteSwap16 (void *d, void *s)
- void byteSwap32 (void *d, void *s)
- void byteSwap64 (void *d, void *s)

Variables

- const BTimeout BTimeoutForever = 0xFFFFFFFF

8.116.1 Macro Definition Documentation

8.116.1.1 BeamlibVersion

```
#define BeamlibVersion 0x030000
```

8.116.2 Typedef Documentation

8.116.2.1 Bool

```
typedef bool Bool
```

8.116.2.2 BInt8

```
typedef int8_t BInt8
```

8.116.2.3 BUInt8

```
typedef uint8_t BUInt8
```

8.116.2.4 BInt16

```
typedef int16_t BInt16
```

8.116.2.5 BUInt16

```
typedef uint16_t BUInt16
```

8.116.2.6 BInt32

```
typedef int32_t BInt32
```

8.116.2.7 BUInt32

```
typedef uint32_t BUInt32
```

8.116.2.8 BInt64

```
typedef int64_t BInt64
```

8.116.2.9 BUInt64

```
typedef uint64_t BUInt64
```

8.116.2.10 BFloat32

```
typedef float BFloat32
```

8.116.2.11 BFloat64

```
typedef double BFloat64
```

8.116.2.12 BChar

```
typedef char BChar
```

8.116.2.13 BInt

```
typedef BInt32 BInt
```

8.116.2.14 BUInt

```
typedef BUInt32 BUInt
```

8.116.2.15 BFloat

```
typedef BFloat32 BFloat
```

8.116.2.16 BDouble

```
typedef BFloat64 BDouble
```

8.116.2.17 BSize

```
typedef size_t BSize
```

8.116.2.18 BArrayFloat

```
typedef std::vector<BFloat32> BArrayFloat
```

8.116.2.19 BArrayDouble

```
typedef std::vector<BFloat64> BArrayDouble
```

8.116.2.20 BTimeout

```
typedef BUInt32 BTimeout
```

8.116.3 Enumeration Type Documentation

8.116.3.1 BEventType

```
enum BEventType
```

Enumerator

BEventTypeNone	
BEventTypeError	
BEventTypeRead	
BEventTypeReadLine	
BEventTypeWrite	
BEventTypeConnect	
BEventTypeDisconnect	
BEventTypeClientConnect	
BEventTypeClientDisconnect	

8.116.3.2 BEventWaitSet

enum BEventWaitSet

Enumerator

BEventWaitNone	
BEventWaitError	
BEventWaitRead	
BEventWaitReadLine	
BEventWaitWrite	
BEventWaitConnect	
BEventWaitDisconnect	
BEventWaitClientConnect	
BEventWaitClientDisconnect	
BEventWaitAny	

8.116.3.3 BType

enum BType

Enumerator

BTypeNone	
BTypeBool	
BTypeInt8	
BTypeUInt8	
BTypeInt16	
BTypeUInt16	
BTypeInt32	
BTypeUInt32	
BTypeInt64	
BTypeUInt64	
BTypeFloat32	
BTypeFloat64	
BTypeChar	
BTypeString	
BTypeError	
BTypeTime	
BTypeTimeUs	
BTypeObj	

8.116.3.4 BTypeComp

enum BTypeComp

Enumerator

BTypeCompSingle	
BTypeCompArray	
BTypeCompArrayFixed	
BTypeCompList	
BTypeCompDict	

8.116.4 Function Documentation

8.116.4.1 timeoutTicks()

```
BTimeout timeoutTicks (
            BTimeout timeoutUs )  [inline]
```

8.116.4.2 byteSwap8()

```
void byteSwap8 (
            void * d,
            void * s )  [inline]
```

8.116.4.3 byteSwap16()

```
void byteSwap16 (
            void * d,
            void * s )  [inline]
```

8.116.4.4 byteSwap32()

```
void byteSwap32 (
            void * d,
            void * s )  [inline]
```

8.116.4.5 byteSwap64()

```
void byteSwap64 (
            void * d,
            void * s )  [inline]
```

8.116.5 Variable Documentation

8.116.5.1 BTimeoutForever

```
const BTimeout BTimeoutForever = 0xFFFFFFFF
```

8.117 BUrl.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <BUrl.h>
#include <curl/curl.h>
```

8.118 BUrl.h File Reference

```
#include <stdio.h>
#include <BString.h>
#include <BError.h>
```

Classes

- class BUrl

  Access to a Url.

8.119 /src/bdev3/beamlib/doc/overview.dox File Reference

Index