

bpipexx
1.0.8_01

Generated by Doxygen 1.8.15

1 bpipexx	1
1.1 Introduction	1
2 Class Index	3
2.1 Class List	3
3 Class Documentation	5
3.1 SAOTrace::BPipeX::DpktF< T > Struct Template Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 DpktF()	6
3.2 SAOTrace::BPipeX::HdrF< T > Struct Template Reference	7
3.2.1 Detailed Description	7
3.2.2 Constructor & Destructor Documentation	8
3.2.2.1 HdrF()	8
3.2.3 Member Function Documentation	8
3.2.3.1 data()	8
3.3 SAOTrace::BPipeX::Simple Class Reference	9
3.3.1 Detailed Description	9
3.3.2 Creating an object and attaching to streams	10
3.3.3 Manipulating Header Fields	10
3.3.4 Manipulating Data Packet Fields	10
3.3.5 Mapping, reading and writing	10
3.3.6 Closing I/O Streams	11
3.3.7 Constructor & Destructor Documentation	11
3.3.7.1 ~Simple()	11
3.3.7.2 Simple() [1/2]	11
3.3.7.3 Simple() [2/2]	11
3.3.8 Member Function Documentation	12
3.3.8.1 add() [1/2]	12
3.3.8.2 add() [2/2]	12
3.3.8.3 delete_dpktf()	13
3.3.8.4 delete_hdrf()	13
3.3.8.5 dpktf_data()	14
3.3.8.6 has_dpktf()	14
3.3.8.7 has_hdrf()	15
3.3.8.8 hdrf_data()	15
3.3.8.9 map()	15
3.3.8.10 read_dpmts()	16
3.3.8.11 write_dpmts()	16

3.3.8.12 write_hdr()	17
--------------------------------	----

Index	19
--------------	-----------

Chapter 1

bpipexx

1.1 Introduction

bpipexx provides C++ sugar for bpipe

- [SAOTrace::BPipeX::Simple](#) provides a simple wrapper around it

Chapter 2

Class Index

2.1 Class List

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

SAOTrace::BPipeX::DpktF< T >	5
SAOTrace::BPipeX::HdrF< T >	7
SAOTrace::BPipeX::Simple	9

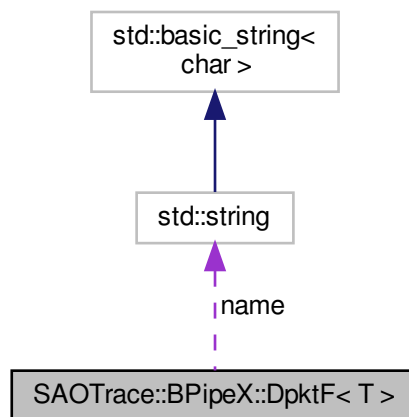
Chapter 3

Class Documentation

3.1 SAOTrace::BPipeX::DpktF< T > Struct Template Reference

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

Collaboration diagram for SAOTrace::BPipeX::DpktF< T >:



Public Member Functions

- `DpktF` (const char *cname)

Construct a scalar field with the given name and type.

Public Attributes

- string **name**
- BMatrix * **matrix**
- BPDataType **BPT**

3.1.1 Detailed Description

```
template<typename T>
struct SAOTrace::BPipeX::DpktF< T >
```

A simple structure to hold information about a \bpipe data field. Note that the existing constructors set the `matrix` field to `NULL`.

Template Parameters

<i>T</i>	The storage type of the data. This is one of the intrinsic types and structures supported by the <code>bpipe</code> library.
----------	--

3.1.2 Constructor & Destructor Documentation

3.1.2.1 DpktF()

```
template<typename T>
SAOTrace::BPipeX::DpktF< T >::DpktF (
    const char * cname )
```

Construct a *scalar* field with the given name and type.

Template Parameters

<i>T</i>	The storage type of the data. This is one of the intrinsic types and structures supported by the <code>bpipe</code> library.
----------	--

Parameters

<i>in</i>	<i>cname</i>	The name of the field.
-----------	--------------	------------------------

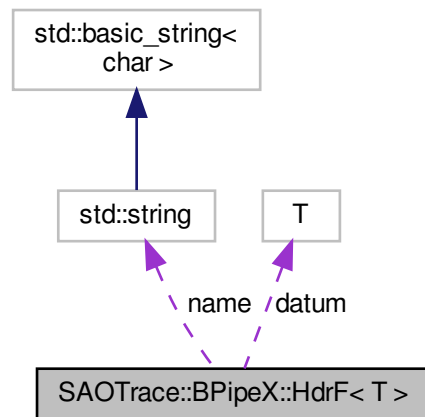
The documentation for this struct was generated from the following file:

- `bpipe.h`

3.2 SAOTrace::BPipeX::HdrF< T > Struct Template Reference

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

Collaboration diagram for SAOTrace::BPipeX::HdrF< T >:



Public Member Functions

- `HdrF` (const char *cname, T idatum)
Construct a scalar field with the given name, data, and type.
- `T * data ()`
Return a pointer to the data.

Public Attributes

- string **name**
- BPMatrix * **matrix**
- BPDataType **BPT**
- T **datum**
- bool **copy**

3.2.1 Detailed Description

```
template<typename T>
struct SAOTrace::BPipeX::HdrF< T >
```

A simple structure to hold information about a `bpipex` header field. Note that the existing constructors set the `matrix` field to `NULL`.

Template Parameters

<i>T</i>	The storage type of the data. This is one of the intrinsic types and structures supported by the <code>bpipe</code> library.
----------	--

3.2.2 Constructor & Destructor Documentation

3.2.2.1 HdrF()

```
template<typename T>
SAOTrace::BPipeX::HdrF< T >::HdrF (
    const char * cname,
    T idatum )
```

Construct a *scalar* field with the given name, data, and type.

Template Parameters

<i>T</i>	The storage type of the data. This is one of the intrinsic types and structures supported by the <code>bpipe</code> library.
----------	--

Parameters

in	<i>cname</i>	The name of the field.
in	<i>idatum</i>	The data to store. This is copied.

3.2.3 Member Function Documentation

3.2.3.1 data()

```
template<typename T>
T* SAOTrace::BPipeX::HdrF< T >::data ( ) [inline]
```

Return a pointer to the data.

Returns

a pointer to the copy of the data

Referenced by `SAOTrace::BPipeX::Simple::add()`.

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

- `bpipexx.h`

3.3 SAOTrace::BPipeX::Simple Class Reference

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

Public Member Functions

- [~Simple](#) ()
Destructor.
- [Simple](#) ()
Constructor.
- [Simple](#) (const string &input, const string &output)
Constructor specifying input and output streams.
- template<typename T >
void [add](#) (HdrF< T > field)
add a header field
- template<typename T >
void [add](#) (const DpktF< T > &field)
add a data packet field
- void [delete_hdrf](#) (const char *name, size_t index=BPHdrfldx_ALL)
delete a header field
- void [delete_dpktf](#) (const char *name, BPDataSite site=BPDSite_OUTPUT, BPipeOutput *channel=BPOutputChannel_ALL)
delete a data packet field
- bool [has_hdrf](#) (const char *name, size_t index=BPHdrfldx_LAST)
test if a header field exists
- bool [has_dpktf](#) (const char *name)
test if a data packet field exists
- template<typename T >
T * [dpktf_data](#) (const char *name)
get a pointer to the data in a data packet field
- template<typename T >
T * [hdrf_data](#) (const char *name, size_t idx=BPHdrfldx_LAST)
get a pointer to the data in a header packet field
- void [map](#) (int npkts=1)
map the data packet fields onto memory.
- size_t [read_dpkts](#) ()
read in the next batch of data packets
- void [write_hdr](#) ()
write the output header
- void [write_dpkts](#) (BPipeOutput *bpo=BPOutputChannel_ALL)
write a data packet
- BPipe * [bpipe](#) ()

3.3.1 Detailed Description

A simple wrapper class providing optimized access to a binary pipe object.

3.3.2 Creating an object and attaching to streams

The easiest means to create an object manipulating bpipe streams is to use the two argument constructor:

```
#include <SAOTrace/bpipexx/bpipexx.h>
using namespace SAOTrace;
BPipeX::Simple bp( "", "stdout" );
```

In this case it opens an output stream, but no input stream, suitable for a ray generator. For a filter, specify both streams, for a sink, specify only the input stream.

For more complex situations, don't specify any streams, and use the `bpipe` accessor to manipulate the underlying `bpipe` object directly:

```
BPipeX::Simple bp();
bpipe_input( bp->bpipe, "stdin" );
```

3.3.3 Manipulating Header Fields

Recall that the header accepts multiple fields with the same name.

The first step is to create a `BPipeX::HdrF` object, then pass it to the `add` method. The steps can be combined:

```
bp.add( BPipeX::HdrF<double>("temp", 33.2 ) );
```

To delete a header field, use `delete_hdrf`:

```
bp.delete_hdrf( "temp" );
```

3.3.4 Manipulating Data Packet Fields

Recall that the data packets can have only one field with a given name.

The first step is to create a `BPipeX::DpktF` object, then pass it to the `add` method. The steps can be combined:

```
bp.add( BPipeX::DpktF<double>("temp") );
```

To delete a data packet field from the output, use `delete_dpktf`:

```
bp.delete_dpktf( "temp" );
```

See the documentation for `delete_dpktf` for more complex use.

3.3.5 Mapping, reading and writing

After all fields are created, the packet images must be mapped:

```
bp.map();
```

To access the location in the core image for a data packet field, use `dpktf_data`:

```
double* temp_p = bp.dpktf_data<double>( "temp" );
```

For filters or sinks, make sure to write the header prior to outputting any data packets:

```
bp.write_hdr();
```

Use `read_dpmts` and `write_dpmts` to read and write.

3.3.6 Closing I/O Streams

The streams will be closed upon destruction of the `BPipeX::Simple` object.

3.3.7 Constructor & Destructor Documentation

3.3.7.1 `~Simple()`

```
SAOTrace::BPipeX::Simple::~~Simple ( )
```

Destructor.

Destroy a `BPipeX::Simple` object, closing the associated output streams.

3.3.7.2 `Simple()` [1/2]

```
SAOTrace::BPipeX::Simple::Simple ( )
```

Constructor.

Construct the underlying `bpipe` object without attaching any streams

3.3.7.3 `Simple()` [2/2]

```
SAOTrace::BPipeX::Simple::Simple (
    const string & input,
    const string & output )
```

Constructor specifying input and output streams.

Either of the parameters may be an empty string, indicating that that stream should not be attached.

Parameters

<code>in</code>	<code>input</code>	the name of the input stream (may be <code>stdin</code>)
<code>in</code>	<code>output</code>	the name of the input stream (may be <code>stdout</code>)

Exceptions

<i>Exception</i>	if it is unable to attach the streams.
------------------	--

3.3.8 Member Function Documentation

3.3.8.1 add() [1/2]

```
template<typename T >
void SAOTrace::BPipeX::Simple::add (
    HdrF< T > field ) [inline]
```

add a header field

For example:

```
bpipe.add( HdrF<double>("temperature", 33.2) );
```

Parameters

in	field	A BPipeX::HdrF object
----	-------	---------------------------------------

Exceptions

<i>Exception</i>	if unable to create the header field
------------------	--------------------------------------

References [SAOTrace::BPipeX::HdrF< T >::data\(\)](#).

3.3.8.2 add() [2/2]

```
template<typename T >
void SAOTrace::BPipeX::Simple::add (
    const DpktF< T > & field ) [inline]
```

add a data packet field

For example:

```
bpipe.add( DpktF<double>("smile") );
```

Parameters

in	field	A BPipeX::DpktF object
----	-------	--

Exceptions

<i>Exception</i>	if unable to create the data packet field or the field exists with a different data type.
------------------	---

3.3.8.3 delete_dpktf()

```
void SAOTrace::BPipeX::Simple::delete_dpktf (
    const char * name,
    BPDataSite site = BPDSite_OUTPUT,
    BPipeOutput * channel = BPOutputChannel_ALL )
```

delete a data packet field

Deletes the specified data packet field. By default the field is removed from the output image of all output channels. Set the `site` parameter to change the data site or the `channel` parameter to specify an alternate output channel

Parameters

in	<i>name</i>	the name of the field to delete
in	<i>site</i>	the <i>optional</i> data site in which to delete it. This may be one of <ul style="list-style-type: none"> • BPDSite_INPUT • BPDSite_CORE, or • BPDSite_OUTPUT
in	<i>channel</i>	<i>optional</i> channel output channel to delete from

Exceptions

<i>Exception</i>	if there is an error
------------------	----------------------

3.3.8.4 delete_hdrf()

```
void SAOTrace::BPipeX::Simple::delete_hdrf (
    const char * name,
    size_t index = BPHdrfIdx_ALL )
```

delete a header field

Deletes the specified header field. By default it will delete all header fields with the given name. Set the `index` parameter to either the index of field to delete, or to BPHdrfIdx_LAST to delete the last one.

Parameters

in	<i>name</i>	the name of the field to delete
in	<i>index</i>	the <i>optional</i> index of the field to delete.

Exceptions

<i>Exception</i>	if there is an error
------------------	----------------------

3.3.8.5 dpktf_data()

```
template<typename T >
T* SAOTrace::BPipeX::Simple::dpktf_data (
    const char * name ) [inline]
```

get a pointer to the data in a data packet field

For example:

```
double* temp_p = bpipe.dpktf_data<double>( "temp" );
```

Template Parameters

<i>T</i>	the type of the stored data.
----------	------------------------------

Parameters

in	<i>name</i>	the name of the data packet field.
----	-------------	------------------------------------

3.3.8.6 has_dpktf()

```
bool SAOTrace::BPipeX::Simple::has_dpktf (
    const char * name )
```

test if a data packet field exists

Parameters

in	<i>name</i>	the name of the field
----	-------------	-----------------------

Returns

true if the field exists

3.3.8.7 has_hdrf()

```
bool SAOTrace::BPipeX::Simple::has_hdrf (
    const char * name,
    size_t index = BPHdrfIdx_LAST )
```

test if a header field exists

Parameters

in	<i>name</i>	the name of the field
in	<i>index</i>	the <i>optional</i> index of the field to delete. Defaults to BPHdrfIdx_LAST

Returns

true if the field exists

3.3.8.8 hdrf_data()

```
template<typename T >
T* SAOTrace::BPipeX::Simple::hdrf_data (
    const char * name,
    size_t idx = BPHdrfIdx_LAST ) [inline]
```

get a pointer to the data in a header packet field

For example:

```
double* temp_p = bpipe.hdrf_data<double>( "temp" );
```

Template Parameters

<i>T</i>	the type of the stored data.
----------	------------------------------

Parameters

in	<i>name</i>	the name of the data packet field.
in	<i>idx</i>	the index of the field. Defaults to the last one.

3.3.8.9 map()

```
void SAOTrace::BPipeX::Simple::map (
    int npkts = 1 )
```

map the data packet fields onto memory.

This function maps the data packet fields onto memory and allocates space for the images.

Parameters

<i>in</i>	<i>npkts</i>	the <i>optional</i> number of packets to allocate space for. defaults to 1
-----------	--------------	--

Exceptions

<i>Exception</i>	if it is unable to allocate the image.
------------------	--

3.3.8.10 read_dpmts()

```
size_t SAOTrace::BPipeX::Simple::read_dpmts ( )
```

read in the next batch of data packets

Read in the number of data packets specified by the `map` method.

Returns

the number of packets read

Exceptions

<i>Exception</i>	if there is an error reading the input stream
------------------	---

3.3.8.11 write_dpmts()

```
void SAOTrace::BPipeX::Simple::write_dpmts (
    BPipeOutput * bpo = BPOutputChannel_ALL )
```

write a data packet

Writes a data packet to one or all output channels.

Parameters

<i>in</i>	<i>bpo</i>	the <i>optional</i> output channel to write to. Defaults to writing to all output channels.
-----------	------------	---

Exceptions

<i>Exception</i>	if there is an error
------------------	----------------------

3.3.8.12 write_hdr()

```
void SAOTrace::BPipeX::Simple::write_hdr ( )
```

write the output header

Writes header information to the output channels

Exceptions

<i>Exception</i>	if there is an error
------------------	----------------------

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

- bpipexx.h
- bpipexx.cc

Index

- ~Simple
 - SAOTrace::BPipeX::Simple, [11](#)
- add
 - SAOTrace::BPipeX::Simple, [12](#)
- data
 - SAOTrace::BPipeX::HdrF< T >, [8](#)
- delete_dpktf
 - SAOTrace::BPipeX::Simple, [13](#)
- delete_hdrf
 - SAOTrace::BPipeX::Simple, [13](#)
- DpktF
 - SAOTrace::BPipeX::DpktF< T >, [6](#)
- dpktf_data
 - SAOTrace::BPipeX::Simple, [14](#)
- has_dpktf
 - SAOTrace::BPipeX::Simple, [14](#)
- has_hdrf
 - SAOTrace::BPipeX::Simple, [14](#)
- HdrF
 - SAOTrace::BPipeX::HdrF< T >, [8](#)
- hdrf_data
 - SAOTrace::BPipeX::Simple, [15](#)
- map
 - SAOTrace::BPipeX::Simple, [15](#)
- read_dpkets
 - SAOTrace::BPipeX::Simple, [16](#)
- SAOTrace::BPipeX::DpktF< T >, [5](#)
 - DpktF, [6](#)
- SAOTrace::BPipeX::HdrF< T >, [7](#)
 - data, [8](#)
 - HdrF, [8](#)
- SAOTrace::BPipeX::Simple, [9](#)
 - ~Simple, [11](#)
 - add, [12](#)
 - delete_dpktf, [13](#)
 - delete_hdrf, [13](#)
 - dpktf_data, [14](#)
 - has_dpktf, [14](#)
 - has_hdrf, [14](#)
 - hdrf_data, [15](#)
 - map, [15](#)
 - read_dpkets, [16](#)
 - Simple, [11](#)
 - write_dpkets, [16](#)
 - write_hdr, [17](#)
- Simple
 - SAOTrace::BPipeX::Simple, [11](#)
- write_dpkets
 - SAOTrace::BPipeX::Simple, [16](#)
- write_hdr
 - SAOTrace::BPipeX::Simple, [17](#)