# SMACQ Reference Manual

Generated by Doxygen 1.3.5

Mon Aug 28 23:36:59 2006

# Contents


1 System for Modular Analysis and Continuous Queries 1

1.1 Embedding SMACQ in Your Application . . . 1

1.2 Creating a SMACQ Type Module . . . 1

1.3 Creating a SMACQ Processing Module . . . 1

1.4 Using a DtsObject . . . 2

2 SMACQ Namespace Index 3

2.1 SMACQ Namespace List . . . 3

3 SMACQ Hierarchical Index 5

3.1 SMACQ Class Hierarchy . . . 5

4 SMACQ Class Index 11

4.1 SMACQ Class List . . . 11

5 SMACQ Namespace Documentation 13

5.1 boost Namespace Reference . . . 13

6 SMACQ Class Documentation 15

6.1 DTS Class Reference . . . 15

6.2 DtsField Class Reference . . . 21

6.3 DtsObject Class Reference . . . 22

6.4 DtsObject_ Class Reference . . . 23

6.5 DtsObjectVec Class Reference . . . 26

6.6 DynamicArray< T > Class Template Reference . . . 27

6.7 FieldVec Class Reference . . . . . 28
6.8 FieldVecElement Class Reference . . . . . 30
6.9 FieldVecHash< T > Class Template Reference . . . . . 31
6.10 FieldVecSet Class Reference . . . . . 32
6.11 Filelist Class Reference . . . . . 33
6.12 FilelistArgs Class Reference . . . . . 34
6.13 FilelistBounded Class Reference . . . . . 35
6.14 FilelistConstant Class Reference . . . . . 37
6.15 FilelistError Class Reference . . . . . 38
6.16 FilelistOneshot Class Reference . . . . . 39
6.17 FilelistStdin Class Reference . . . . . 40
6.18 PthreadMutex Class Reference . . . . . 41
6.19 RecursiveLock Class Reference . . . . . 43
6.20 SmacqGraph Class Reference . . . . . 44
6.21 SmacqGraphContainer Class Reference . . . . . 50
6.22 SmacqModule Class Reference . . . . . 53
6.23 SmacqModule::algebra Struct Reference . . . . . 56
6.24 SmacqModule::smacq_init Struct Reference . . . . . 57
6.25 SmacqScheduler Class Reference . . . . . 58
6.26 StrucioStream Class Reference . . . . . 61
6.27 StrucioWriter Class Reference . . . . . 63
6.28 ThreadedSmacqModule Class Reference . . . . . 65
6.29 ThreadSafeBoolean Class Reference . . . . . 69
6.30 ThreadSafeContainer< T, CONTAINER > Class Template Reference 70
6.31 ThreadSafeCounter Class Reference . . . . . 72
6.32 ThreadSafeDynamicArray< T > Class Template Reference . . . . 73
6.33 ThreadSafeMap< KEY, VALUE, LT > Class Template Reference . 74
6.34 ThreadSafeMultiSet< T > Class Template Reference . . . . . 75
6.35 ThreadSafeRandomAccessContainer< T, CONTAINER > Class Template Reference . . . . . 77
6.36 ThreadSafeStack< T > Class Template Reference . . . . . 79

6.37 ThreadSafeVector< T > Class Template Reference . . . . . 80

# Chapter 1

# System for Modular Analysis and Continuous Queries

Version:
2.5

SMACQ is an extensible system for analyzing streams of structured data.

## 1.1 Embedding SMACQ in Your Application

All work is done in one or more SmacqGraph instances. The easiest way to construct a SmacqGraph is with SmacqGraph::newQuery(). To execute a graph, you also need to instantiate a SmacqScheduler and use its methods like SmacqScheduler::input(), SmacqScheduler::busy_loop(), etc.

## 1.2 Creating a SMACQ Type Module

Type modules define interfaces for parsing data. See the dts-modules manpage for more information.

## 1.3 Creating a SMACQ Processing Module

A data-processing module should be a subclass of SmacqModule or ThreadedSmacqModule.

## 1.4 Using a DtsObject

SMACQ uses the DtsObject abstraction for all data that it handles. C++ programmers should ALWAYS use a DtsObject auto-pointer rather than referencing the underlying DtsObject_ class directly. Classes like FieldVecHash and FieldVecSet are provided to make it easier to do common tasks with DtsObject.

# Chapter 2

# SMACQ Namespace Index

## 2.1 SMACQ Namespace List

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

boost (We overload the default == operator for DtsObject so that it compares the objects instead of the pointers to the objects ) . . . . . 13.

# Chapter 3

# SMACQ Hierarchical Index

## 3.1 SMACQ Class Hierarchy

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


_dts_msg
_ExportPktData
_list_item
_patricia_node_t
_patricia_tree_t
_pre x4_t
_pre x_t
_smacq_opt
alias
arglist
base_year_is_a_multiple_of_100
batch
Bloom< T >
BloomCounters< T, COUNTER>
BloomSet< T >
bzFile_t
cmalloc
cmalloc_element
DatalogNamespace
dfa
dfa_state
dts_arith_operand
dts_comp_func
dts_comparison
dts_ eld_info

dts_ eld_spec
dts_list
dts_literal_operand
dts_operand
dts_pkthdr
dts_type
dts_type_info
DtsField . . . . . 21
DtsObject . . . . . 22
DtsObjectVec . . . . . 26
DynamicArray< T > . . . . . 27
SmacqModule::UsesArray
ThreadSafeContainer< T, DynamicArray< T > > . . . . . 70
ThreadSafeRandomAccessContainer< T, DynamicArray< T > > . . 77
ThreadSafeDynamicArray< T > . . . . . 73
DynamicArray< bool >
std::equal_to< char * >
extended_pkthdr
elds
FieldVec . . . . . 28
FieldVecBloomCounters
FieldVecBloomSet
FieldVecElement . . . . . 30
FieldVecHash< T > . . . . . 31
FieldVecSet . . . . . 32
Filelist . . . . . 33
FilelistArgs . . . . . 34
FilelistBounded . . . . . 35
FilelistConstant . . . . . 37
FilelistError . . . . . 38
FilelistOneshot . . . . . 39
FilelistStdin . . . . . 40
ags< N, T >
get_line
GraphVector
group
__gnu_cxx::hash< DtsObjectVec >
hash_map
ldMap< T >
join
joinlist
std::less< FieldVec >
list
list_element
ltstr

map
map
mask
MethodFunctor< Return, Arg, Class, Callback
obj_list
old_pcap_pkthdr
parser_control
pattern
per_dimension_state
per_id_dimension_state
per_id_state
PerType
pickle
PthreadMutex . . . . . 41
DTS . . . . . 15
DtsObject_ . . . . . 23
runq
SmacqGraph . . . . . 44
ThreadSafeContainer< T, CONTAINER > . . . . . 70
ThreadSafeRandomAccessContainer< T, CONTAINER > . . . . . 77
ThreadSafeMultiSet< T > . . . . . 75
ThreadSafeVector< T > . . . . . 80
ThreadSafeVector< ThreadSafeMultiSet< SmacqGraph_ptr > > 80
Children
ThreadSafeRandomAccessContainer< T, std::vector< T > > . . . . . 77
ThreadSafeRandomAccessContainer< ThreadSafeMultiSet< SmacqGraph_ptr >, std::vector< ThreadSafeMultiSet< SmacqGraph_ptr > > . . . . . 77
ThreadSafeContainer< T, DynamicArray< T > > . . . . . 70
ThreadSafeContainer< T, std::vector< T > > . . . . . 70
ThreadSafeContainer< ThreadSafeMultiSet< SmacqGraph_ptr >, std::vector< ThreadSafeMultiSet< SmacqGraph_ptr > > . . 70
ThreadSafeMap< KEY, VALUE, LT > . . . . . 74
ThreadSafeStack< T > . . . . . 79
PyDtsObject
PySmacqObject
RecursiveLock . . . . . 43
re ist_element
relative_time
ruleset
runq< T >::qel
sa_alignof_helper< type >
smacq_functions
smacq_options
smacq_optval

SmacqGraphContainer . . . . . 50
SmacqModule . . . . . 53
ThreadedSmacqModule . . . . . 65
ThreadedSmacqModule . . . . . 65
SmacqModule::algebra . . . . . 56
SmacqModule::smacq_init . . . . . 57
SmacqScheduler . . . . . 58
sockdatum
srcstat
SrcStat
stack
StackException
state
StrucioStream . . . . . 61
FileStream
StrucioWriter . . . . . 63
StrucioPcapWriter
substr_search_result
substr_search_resume
table
tcphdr
TermList
textint
ThreadSafeBoolean . . . . . 69
ThreadSafeCounter . . . . . 72
time_t_is_integer
timespec
timeval_32
transition
trie
trieobj
twos_complement_arithmetic
vector
vector
ThreadSafeContainer< T, std::vector< T > > . . . . . 70
vector
vector
ThreadSafeContainer< ThreadSafeMultiSet< SmacqGraph_ptr >, std::vector< ThreadSafeMultiSet< SmacqGraph_ptr > > > . . 70
vector
vector
vector
vector
DynamicArray< bool > . . . . . 27
vector

vector
vector
vector
vlan_hdr
vphrase
yy_buffer_state
yyalloc
yystype
YYSTYPE

# Chapter 4

# SMACQ Class Index

## 4.1 SMACQ Class List

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


DTS (DTS is a Dynamic Type System run-time environment ) . . . . . 15
DtsField (DtsField is a vector class used to describe a field specification such as foo.bar.baz translated into numeric identifiers for fast lookup ) 21
DtsObject (A DtsObject is an auto-pointer to a DtsObject_ instance ) . . . . . 22
DtsObject_ (DtsObject_ instances should only be used via DtsObject auto-pointers (the auto-pointer keeps track of reference counts for the user) ) . . . . . 23
DtsObjectVec (A vector of DtsObject elements ) . . . . . 26
DynamicArray< T > (This is a wrapper around vector which grows the array as necessary to satisfy [] operations ) . . . . . 27
FieldVec (A vector of fields from a DtsObject) . . . . . 28
FieldVecElement (An element of a FieldVec) . . . . . 30
FieldVecHash< T > (A hash_map (table) for FieldVec) . . . . . 31
FieldVecSet (A hash_set for FieldVec) . . . . . 32
Filelist (A pure virtual base for classes that return filenames ) . . . . . 33
FilelistArgs (Return file names from an argument vector ) . . . . . 34
FilelistBounded (Return filenames from an index file ) . . . . . 35
FilelistConstant (Return a single filename ) . . . . . 37
FilelistError (Never return a file name ) . . . . . 38
FilelistOneshot (Return a single filename ) . . . . . 39
FilelistStdin (Return file names from STDIN ) . . . . . 40
PthreadMutex (This is a Mutex that can be acquired multiple times, recursively, by the same thread without causing deadlock ) . . . . . 41
RecursiveLock (On instantiation, this class will lock a PthreadMutex and unlock it automatically upon destruction ) . . . . . 43

SmacqGraph (This is a node in a graph. Each node references its parents and children ) . . . . . 44
SmacqGraphContainer (This is a container for an SmacqGraph which may have multiple heads or tails ) . . . . . 50
SmacqModule (A virtual base class for SMACQ modules ) . . . . . 53
SmacqModule::algebra (The algebra element is optional and is used only by the data ow optimizer ) . . . . . 56
SmacqModule::smacq_init (This context structure is passed to Smacq-Module constructors ) . . . . . 57
SmacqScheduler (This is a scheduler for processing any number of Smacq-GraphContainer instances ) . . . . . 58
StrucioStream (Pure-virtual base class for input streams ) . . . . . 61
StrucioWriter (A le writer for structured data ) . . . . . 63
ThreadedSmacqModule (A virtual base clase for SMACQ modules that are executed with in their own "thread" instead of being event-driven ) . . . . . 65
ThreadSafeBoolean (A thread-safe boolean value ) . . . . . 69
ThreadSafeContainer< T, CONTAINER > (A base class for thread-safe containers ) . . . . . 70
ThreadSafeCounter (An atomic, thread-safe counter ) . . . . . 72
ThreadSafeDynamicArray< T > (A thread-safe dynamic array ) . . . . . 73
ThreadSafeMap< KEY, VALUE, LT > (A Thread-safe map based on an STL map ) . . . . . 74
ThreadSafeMultiSet< T > (A thread-safe multiset ) . . . . . 75
ThreadSafeRandomAccessContainer< T, CONTAINER > (A base class for random-access thread-safe containers ) . . . . . 77
ThreadSafeStack< T > (A thread-safe version of the STL stack class ) . . . 79
ThreadSafeVector< T > (A thread-safe version of the STL vector class ) . . 80

# Chapter 5

# SMACQ Namespace Documentation

## 5.1 boost Namespace Reference

### 5.1.1 Detailed Description

We overload the default == operator for DtsObject so that it compares the objects instead of the pointers to the objects.

### Functions

- template<> bool operator==< DtsObject_, DtsObject_> (const DtsObject &x, const DtsObject &y)

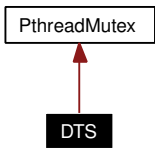
# Chapter 6

# SMACQ Class Documentation

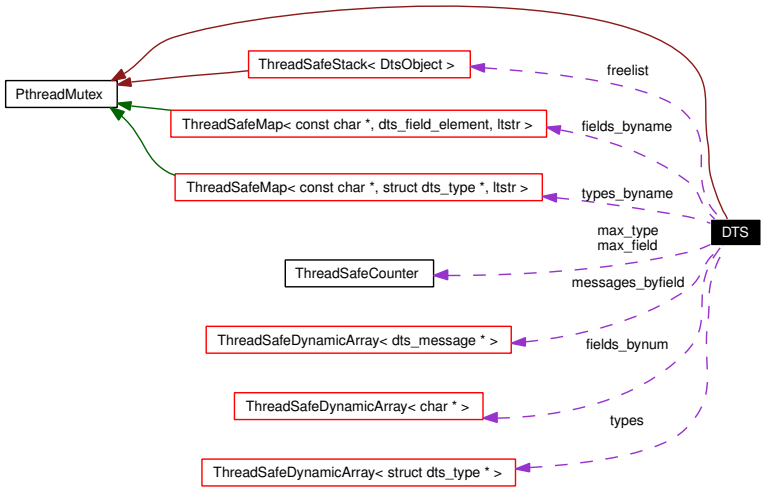
## 6.1 DTS Class Reference

#include < dts.h >

Inheritance diagram for DTS:




Collaboration diagram for DTS:




### 6.1.1 Detailed Description

DTS is a Dynamic Type System run-time environment.

You probably only want one instance of the DTS for your entire program. Factory methods are used to construct DtsObjects, which are typed using the DTS.

## Public Member Functions

- DTS ()

  Construct a new DTS.

- int fromstring (dts_typeid, const char datastr, DtsObject data)
- int dts_lt (int type, void p1, int len1, void p2, int len2)
- void send_message (DtsObject, dts_ eld_element, dts_comparison)

  Send the given object to the given eld of all objects that satisfy the comparison.

- DtsObject msg_check (DtsObject o, dts_ eld_element eld)

  Check for a message for this obejct and eld.

- void use_master ()

  Make eld and type values the same as a master process.

### Factory Methods

- DtsObject construct(dts_typeid, void data)
  Construct a new object with a copy of the given data.
- DtsObject construct_fromstring(dts_typeid type, const char data)
  Construct a new object with data parsed from the given string.
- DtsObject newObject(dts_typeid)
  Return a new object of the given type.
- DtsObject newObject(dts_typeid, int size)
  Return a new object of the given type and size.
- DtsObject readObject (struct pickle pickle, int fd)

Field IDs

DtsObjects expose 0 or more fields (attributes) that can be accessed.

Each field is assigned a numeric identifier, a DtsField, specific to this runtime environment. Fields names can be nested (e.g. "foo.bar.baz") which translates to nested numeric IDs (e.g. "1.3.2"). It is recommended for performance that modules convert type names to IDs sparingly and cache results.

- DtsField require field (char name)
  Convert the given field name into a numeric identifier.
- char field_getname(DtsField &f)
  Return the name of the specified field.
- char field_getname(dts_field_element f)
  Return the name of the specified field.

Type IDs

Types are dynamically loaded classes.

Each type is assigned a numeric identifier specific to this runtime environment. All DtsObjects are typed with these values. It is recommended for performance that modules convert type names to IDs sparingly and cache results.

- dts_typeid requiretype(const char name)
  Load the specified type module (if it is not already loaded) and return the dynamically assigned numeric identifier for that type.
- dts_typeid typenum_byname(const char name)
  If the specified type module is already loaded, this result is the same as requiretype().

- char typename_bynum(const dts_typeid)
  Return the name of the given type.
- dts_type type_bynum(const dts_typeid id)
  Return the type structure for the given type.
- int type_size(const dts_typeid type)
  Return the size (in bytes) of the speci ed type.

Interface to data testing system

- int parsetest(dts_comparison comp, char test)
- int match (DtsObject datum, dts_comparison comps)
- dts_comparison parse_tests(int argc, char argv)
- dts_operand parse_expr(int argc, char argv)

Warnings

- void set_no_warnings()
  Don't warn.
- bool warn_missing_ elds()
  Get current setting.

## Public Attributes

- ThreadSafeStack< DtsObject > freelist
  This freelist should only be used by the DtsObject implementation.

## Friends

- int yysmacql_parse()
- int yydatalogparse()
- int yy lterparse ()
- int yyexprparse()

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 DTS::DTS ()

Construct a new DTS.

You probably only want your program to use pointers to a single instance for the whole program.

### 6.1.3 Member Function Documentation

#### 6.1.3.1 DtsObject DTS::construct (dts_typeid, void data)

Construct a new object with a copy of the given data.

The amount of data copied is determined by the requested typeid.

#### 6.1.3.2 DtsObject DTS::construct_fromstring (dts_typeid type, const char data)

Construct a new object with data parsed from the given string.

The input string should be format accordingly for the given typeid.

#### 6.1.3.3 dts_typeid DTS::requiretype (const char name)

Load the speci ed type module (if it is not already loaded) and return the dynamically assigned numeric identi er for that type.

Didn't already exist, so do it the hard way

#### 6.1.3.4 int DTS::type_size (const dts_typeid type) [inline]

Return the size (in bytes) of the speci ed type.

-1 if size is variable, -2 if type doesn't exist.

#### 6.1.3.5 int DTS::typenum_byname (const char name) [inline]

If the speci ed type module is already loaded, this result is the same as requiretype().

Unlike requiretype(), if the type is not loaded, -1 is returned.

### 6.1.4 Member Data Documentation

#### 6.1.4.1 ThreadSafeStack< DtsObject > DTS::freelist

This freelist should only be used by the DtsObject implementation.

When an object is freed, it can be put on the freelist instead of being destroyed. DtsObject() will use objects on the freelist before constructing new objects.

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

- dts.h

- libsmacq.cpp
- DTS.cpp
- parsing.cpp
- pickle.cpp

## 6.2 DtsField Class Reference

#include < DtsField.h >

### 6.2.1 Detailed Description

DtsField is a vector class used to describe a field specification such as foo.bar.baz translated into numeric identifiers for fast lookup.

### Public Member Functions

- DtsField (dts_field_element fe)
- operator bool () const
- bool operator! () const

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

- DtsField.h

## 6.3 DtsObject Class Reference

#include < DtsObject.h >

### 6.3.1 Detailed Description

A DtsObject is an auto-pointer to a DtsObject_ instance.

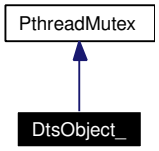
The documentation for this class was generated from the following file:

- DtsObject.h

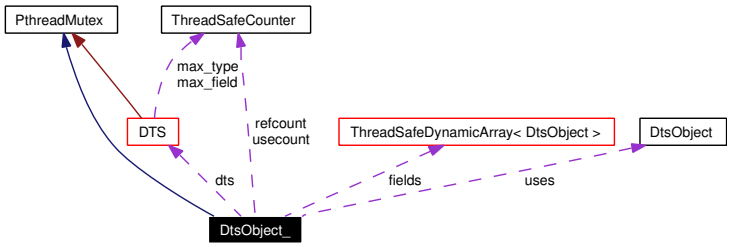
# 6.4 DtsObject_ Class Reference

#include < DtsObject.h >

Inheritance diagram for DtsObject_:




Collaboration diagram for DtsObject_:




## 6.4.1 Detailed Description

DtsObject_ instances should only be used via DtsObject auto-pointers (the auto-pointer keeps track of reference counts for the user).

An object is read-only except for initializaiton, when it is assumed to only have a single user (and therefore not require locking.). Only the eld cache is locked for thread safety.

## Reference Counting

Programmers should NOT use these methods directly

- void intrusive_ptr_add_ref(DtsObject_ o)

  Used by DtsObject (boost::intrusive_ptr).

- void intrusive_ptr_release(DtsObject_ o)

  Used by DtsObject (boost::intrusive_ptr).

## Public Member Functions

- DtsObject_ (DTS dts, int size, int type)
- void init (int size, dts_typeid type)

  (Re-)initialize the object to the given size and type

- DtsObject make_writable ()
- void prime_all_ elds ()
- void prime_ eld (dts_ eld_info )
- std::vector< DtsObject> eldcache()

  Get a copy of the entire eld cache.

- int write (struct pickle pickle, int fd)
- void send(dts_ eld_element eldnum, dts_comparison comparisons)
- void send(DtsField& eld, dts_comparison comparisons)
- int match (dts_comparison comps)
- double eval_arith_operand(struct dts_operand op)

  Expr module uses this.

- DTS getDts() const

  Pointer to the DTS used by this type.

### Copy Constructors

- DtsObject dup ()

  Return a new object with a copy of the data and a private eld vector.

- DtsObject private_copy()

  Return a new object with shared data, but a private eld vector.

### Meta-data Methods

- void setsize(int size)
- int getsize() const
- unsigned long getid () const
- unsigned char getdata() const
- dts_typeid gettype() const
- void settype(int type)

### Initializers

- void setdata(void data)
- void setdatacopy(const void src)

- int set_fromstring (const char datastr)

Field Access

- DtsObject get eld (DtsField & eldv, bool nowarn=false)
  Return a eld object.
- DtsObject get eld (char s, bool nowarn=false)
  Less ef cient lookup by string.
- void attach_ eld (DtsField & eld, DtsObject eld_data)

## Friends

- DtsObject DTS::msg_check(DtsObject, dts_ eld_element)

### 6.4.2 Member Function Documentation

#### 6.4.2.1 DtsObject DtsObject_::dup ()

Return a new object with a copy of the data and a private eld vector.

The eld vector is a copy of the original.

#### 6.4.2.2 DtsObject DtsObject_::private_copy ()

Return a new object with shared data, but a private eld vector.

The eld vector is a copy of the original.

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

- DtsObject.h
- libsmacq.cpp
- DtsObject.cpp
- libsmacq/ lter.cpp
- pickle.cpp

## 6.5 DtsObjectVec Class Reference

#include < FieldVec.h >

### 6.5.1 Detailed Description

A vector of DtsObject elements.

### Public Member Functions

- DtsObjectVec (FieldVec &v)
- DtsObjectVec (DtsObject &o)
- size_t hash (int seed=0) const

  Hash into value in [0..range].

- bool masks (const DtsObjectVec &b) const

  Return true iff argument vector mask ts this vector.

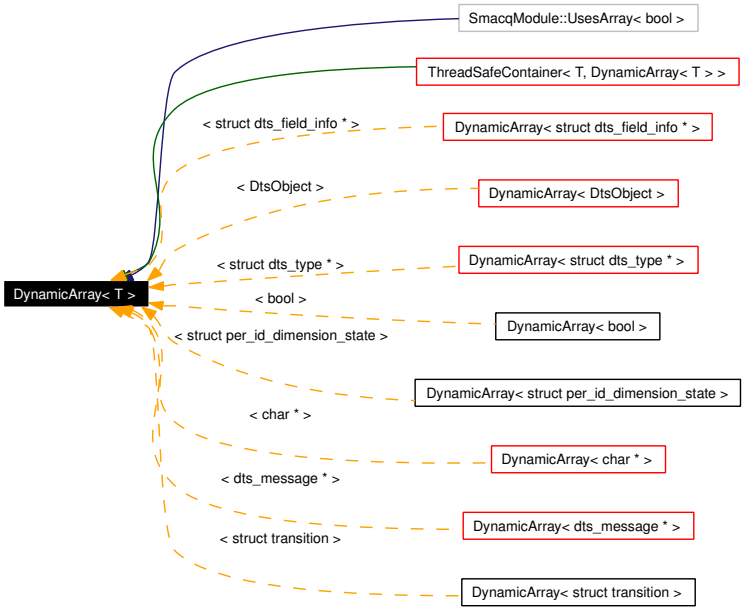
The documentation for this class was generated from the following le:

- FieldVec.h

# 6.6 DynamicArray< T > Class Template Reference

#include < DynamicArray.h >

Inheritance diagram for DynamicArray< T >:




## 6.6.1 Detailed Description

template< class T> class DynamicArray< T >

This is a wrapper around vector which grows the array as necessary to satisfy [] operations.

## Public Member Functions

- T & operator[] (const unsigned int x)

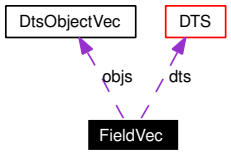
The documentation for this class was generated from the following file:

- DynamicArray.h

## 6.7 FieldVec Class Reference

#include < FieldVec.h >

Collaboration diagram for FieldVec:




### 6.7.1 Detailed Description

A vector of elds from a DtsObject.

### Public Member Functions

- bool get elds (DtsObject o)

  Initialize elds from this DtsObject. Return true if one or more elds present.

- bool has_unde ned () const

  Return true iff one of the vector elements is unde ned for this DtsObject.

- void init (DTS , int argc, char argv)

  Initialize eld vector from an argument vector.

- FieldVec()

  Construct an empty vector. Use init() to initialize later.

- FieldVec(DTS dts, int argc, char argv)

  Construct and initialize eld vector from an argument vector.

- DtsObjectVec& getobjs()

  Return a copy of the vector of current objects.

- const DtsObjectVec& getobjs() const
- bool operator== (const FieldVec &b) const

## 6.7.2 Member Function Documentation

### 6.7.2.1 bool FieldVec::has_undefined () const [inline]

Return true iff one of the vector elements is undefined for this Object

Recomputed when get fields() is called.

### 6.7.2.2 void FieldVec::init (DTS , int argc, char argv) [inline]

Initialize field vector from an argument vector.

Deletes any previous contents.

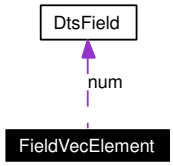
The documentation for this class was generated from the following file:

- FieldVec.h

## 6.8 FieldVecElement Class Reference

#include < FieldVec.h >

Collaboration diagram for FieldVecElement:




### 6.8.1 Detailed Description

An element of a FieldVec.

### Public Attributes

- char name
- DtsField num

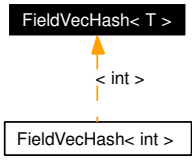
The documentation for this class was generated from the following le:

- FieldVec.h

# 6.9 FieldVecHash< T > Class Template Reference

#include < FieldVec.h >

Inheritance diagram for FieldVecHash< T > :




## 6.9.1 Detailed Description

template< class T> class FieldVecHash< T >

A hash_map (table) for FieldVec

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

- FieldVec.h

## 6.10 FieldVecSet Class Reference

#include < FieldVec.h >

### 6.10.1 Detailed Description

A hash_set for FieldVec

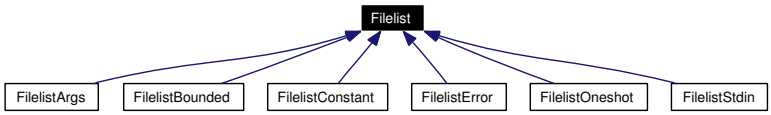
The documentation for this class was generated from the following file:

- FieldVec.h

# 6.11 Filelist Class Reference

#include < Filelist.h >

Inheritance diagram for Filelist:




## 6.11.1 Detailed Description

A pure virtual base for classes that return lenames.

## Public Member Functions

- virtual void next lename (char buf, int len)=0

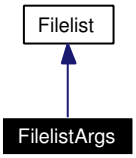
The documentation for this class was generated from the following le:

- Filelist.h

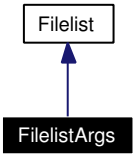
## 6.12 FilelistArgs Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistArgs:




Collaboration diagram for FilelistArgs:




### 6.12.1 Detailed Description

Return le names from an argument vector.

### Public Member Functions

- FilelistArgs (int, char )
- void next lename (char , int)

### Protected Attributes

- int strucio_argc
- char strucio_argv

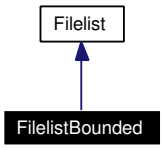
The documentation for this class was generated from the following le:

- Filelist.h

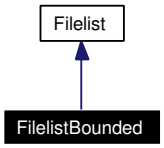
# 6.13 FilelistBounded Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistBounded:




Collaboration diagram for FilelistBounded:




## 6.13.1 Detailed Description

Return lenames from an index le.

## Public Member Functions

- FilelistBounded (char root, long long lower, long long upper)
- void next lename(char , int)

## Protected Attributes

- char index le
- FILE index_fh
- long longlower_bound
- long longupper_bound

## 6.13.2 Member Function Documentation

### 6.13.2.1 void FilelistBounded::next lename (char le , int len) [inline, virtual]

XXX: unchecked size

Implements Filelist.

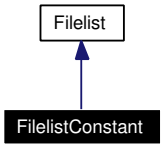
The documentation for this class was generated from the following file:

- Filelist.h

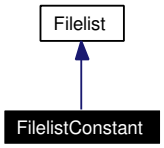
## 6.14 FilelistConstant Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistConstant:




Collaboration diagram for FilelistConstant:




### 6.14.1 Detailed Description

Return a single lename.

### Public Member Functions

- FilelistConstant (char lename)
- void next lename (char lename, int len)

### Protected Attributes

- char le

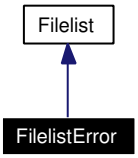
The documentation for this class was generated from the following le:

- Filelist.h

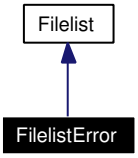
## 6.15 FilelistError Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistError:




Collaboration diagram for FilelistError:




### 6.15.1 Detailed Description

Never return a le name.

### Public Member Functions

- void next lename (char buf, int len)

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

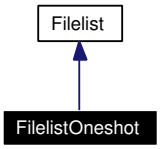
- Filelist.h

## 6.16 FilelistOneshot Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistOneshot:


Collaboration diagram for FilelistOneshot:




### 6.16.1 Detailed Description

Return a single lename.

### Public Member Functions

- FilelistOneshot(char lename)
- void next lename (char lename, int len)

### Protected Attributes

- char le

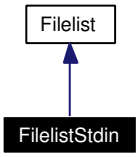
The documentation for this class was generated from the following le:

- Filelist.h

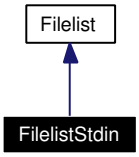
## 6.17 FilelistStdin Class Reference

#include < Filelist.h >

Inheritance diagram for FilelistStdin:




Collaboration diagram for FilelistStdin:




### 6.17.1 Detailed Description

Return le names from STDIN.

### Public Member Functions

- void next lename (char , int)

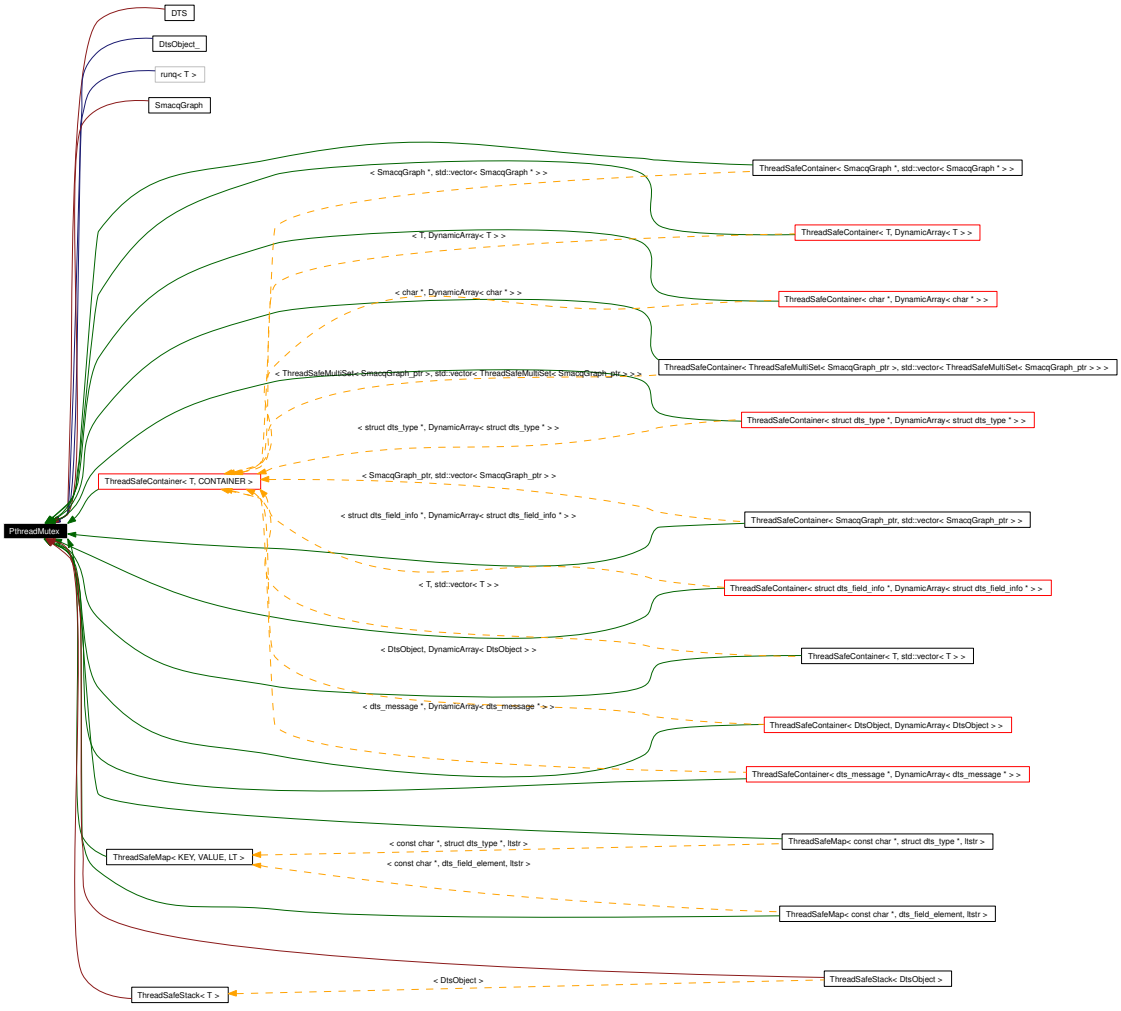
The documentation for this class was generated from the following le:

- Filelist.h

# 6.18 PthreadMutex Class Reference

#include < ThreadSafe.h >

Inheritance diagram for PthreadMutex:




## 6.18.1 Detailed Description

This is a Mutex that can be acquired multiple times, recursively, by the same thread without causing deadlock.

## Protected Member Functions

- void lock ()
- bool try_lock ()
- void unlock ()

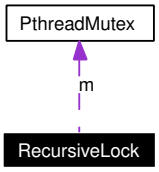
The documentation for this class was generated from the following file:

- ThreadSafe.h

# 6.19 RecursiveLock Class Reference

#include < ThreadSafe.h >

Collaboration diagram for RecursiveLock:




## 6.19.1 Detailed Description

On instantiation, this class will lock a PthreadMutex and unlock it automatically upon destruction.

## Public Member Functions

- RecursiveLock(PthreadMutex _m)
- RecursiveLock(PthreadMutex &_m)

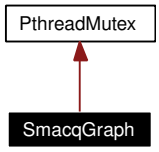
The documentation for this class was generated from the following file:

- ThreadSafe.h

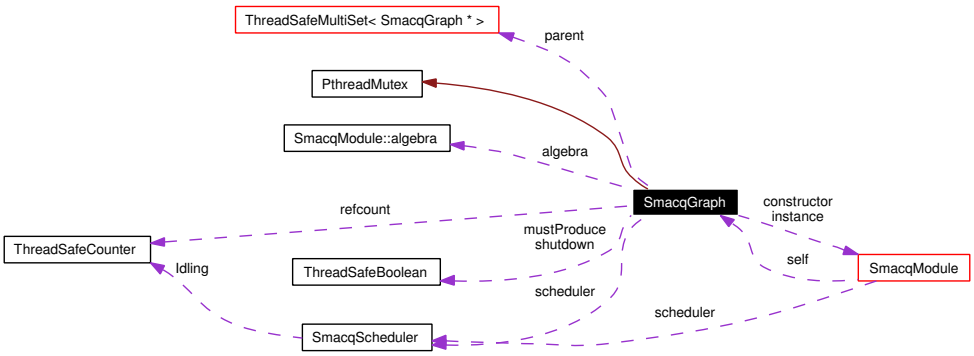
## 6.20 SmacqGraph Class Reference

#include < SmacqGraph.h >

Inheritance diagram for SmacqGraph:




Collaboration diagram for SmacqGraph:




### 6.20.1 Detailed Description

This is a node in a graph. Each node references its parents and children.

### Parent/Child Relationships

- void join (SmacqGraph)

  Attach the speci ed graph onto the tail(s) of the graph(s).

- void join (SmacqGraphContainer, bool dofree=false)

  Attach the speci ed graph onto the tail(s) of the graph(s).

- void replace(SmacqGraphContainer)

Modify parent(s) and children to replace myself with the speci ed graph.

- void dynamic_insert(SmacqGraph, DTS )

  Insert a new graph between my parents and me.

- void add_child(SmacqGraph_ptr child, unsigned int channel=0)

  Add a new graph as one of my children.

- void add_children(SmacqGraphContainer child, unsigned int channel=0)

  Establish a parent/child relationship on the speci ed channel.

- void remove_parent(SmacqGraph parent)

  Remove the speci ed graph from the list of this graph's parents.

- void remove_child_bynum(int, int)
- void remove_child(SmacqGraph_ptr)
- void replace_child(int, int, SmacqGraph newchild)
- void replace_child(int, int, SmacqGraphContainer newchild)
- void replace_child(SmacqGraph oldchild, SmacqGraph newchild)
- void replace_child(SmacqGraph oldchild, SmacqGraphContainer newchild)

- void remove_children()
- bool live_children ()
- bool live_parents()
- const std::vector< ThreadSafeMultiSet< SmacqGraph_ptr > > getChildren ()
- void move_children(SmacqGraph from, SmacqGraph to, bool addvector=false)

## Factories

- SmacqGraph new_child(int argc, char argv)

  Construct a new graph using the given arguments.

- SmacqGraph clone(SmacqGraph newParent)

  Recursively clone a graph.

- SmacqGraphContainer newQuery(DTS , SmacqScheduler, int argc, char argv)

  Parse a query and construct a new graph to execute it.

## Public Member Functions

- bool set(int argc, char argv)
  (Re-)Initialize module
- const int getArgc() const
  Return argc.
- char const getArgv() const
  Return argv (do not modify).
- SmacqGraph(int argc, char argv)
- SmacqGraph init (DTS , SmacqScheduler)
  This method must be called before a graph is used.
- void print (FILE fh, int indent)
  Print the graph.
- std::string print_query()
  Print the graph in re-parsable syntax.
- void log (const char format,...)
  Log something about this graph (printf-style arguments).
- bool distribute_children(DTS )
  Attempt to distribute children of this graph. Return true iff successful.

### Invariant Optimization

- SmacqGraph getInvariants(DTS , SmacqScheduler, DtsField&)
  Return a subgraph containing only invariants over the speci ed eld.
- SmacqGraph getChildInvariants(DTS , SmacqScheduler, DtsField&)
  Same as getInvariants() but operates only on the graph's children.

### Scheduling

- void seed_produce()
  Schedule the node to produce.
- void runable(DtsObject)
  Schedule the given object to be consumed.

- void produce_done()
  Scheduler is done handling a mustProduce.

### Static Public Member Functions

- void do_shutdown(SmacqGraph_ptr f)
  Shutdown a graph node (will propagate to parents and children).

### Public Attributes

- runq< DtsObject > inputq
  Queue of input items to consume.

### Protected Attributes

- char argv
- int argc
- SmacqModule::algebra algebra
- ThreadSafeBoolean shutdown
- ThreadSafeBoolean mustProduce
- SmacqModule instance

### Friends

- void intrusive_ptr_add_ref (SmacqGraph o)
- void intrusive_ptr_release(SmacqGraph o)
  Decrement the reference count.

## 6.20.2 Member Function Documentation

### 6.20.2.1 SmacqGraph SmacqGraph::clone (SmacqGraph newParent)

Recursively clone a graph.

The clone is made a child of newParent, unless newParent is NULL.

6.20.2.2 void SmacqGraph::do_shutdown (SmacqGraph_ptr) [static]

Shutdown a graph node (will propagate to parents and children).

The node may be destroyed by this call.

6.20.2.3 SmacqGraph SmacqGraph::getInvariants (DTS , SmacqScheduler , DtsField &)

Return a subgraph containing only invariants over the specified field.

The subgraph will contain only stateless filters that are applied to all objects in the graph (e.g. not within an OR) and that do NOT use the specified field. The returned graph is newly allocated.

6.20.2.4 SmacqGraph SmacqGraph::init (DTS , SmacqScheduler )

This method must be called before a graph is used.

The graph may be modified as a side-effect, so the caller should replace the called object with the return pointer.

6.20.2.5 void SmacqGraph::move_children (SmacqGraph from, SmacqGraph to, bool addvector= false) [inline, static]

6.20.2.6 SmacqGraph SmacqGraph::new_child (int argc, char argv)

Construct a new graph using the given arguments.

The new graph is automatically attached as a child of the current graph.

## 6.20.3 Friends And Related Function Documentation

6.20.3.1 void intrusive_ptr_release (SmacqGraph o) [friend]

Decrement the reference count.

If the refcount is 0, then clean-up references and destroy

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

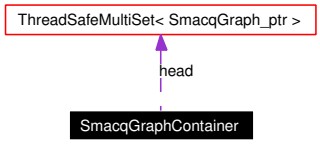
- SmacqGraph.h
- libsmacq.cpp
- optimize.cpp
- parsing.cpp

- SmacqGraph.cpp

## 6.21 SmacqGraphContainer Class Reference

#include < SmacqGraph.h >

Collaboration diagram for SmacqGraphContainer:




### 6.21.1 Detailed Description

This is a container for a SmacqGraph which may have multiple heads or tails.

### Public Member Functions

- SmacqGraphContainer()
  Default CTOR.
- SmacqGraphContainer(ThreadSafeMultiSet< SmacqGraph_ptr > &children)
  Construct from a vector of Children.
- void init (DTS , SmacqScheduler, bool do_optimize=true)
  This method must be called before the graphs are used.
- void shutdown()
  Shutdown all graphs.
- void clear()
  Erase container.
- void print (FILE fh, int indent)
  Print the graphs.
- std::string print_query()
  Print the graph in re-parsable syntax.
- SmacqGraphContainer clone(SmacqGraph newParent)
  Recursively clone a graph.

- void add_clone (SmacqGraph_ptr x, SmacqGraph newParent)
  Add a clone of a graph to this container.
- SmacqGraph getInvariants (DTS , SmacqScheduler, DtsField &)
  Return a subgraph containing only invariants over the speci ed eld.
- void optimize ()
  Preoptimize graph (unnecessary after init).

Combining Graphs

A container can have multiple heads and tails and may even be disconnected.

- void join (SmacqGraph)
  Attach the speci ed graph onto the tail(s) of the graph(s).
- void join (SmacqGraphContainer, bool dofree=false)
  Attach the speci ed graph onto the tail(s) of the graph(s).
- void add_graph (SmacqGraph)
  Add a new graph head.
- void add_graph (SmacqGraphContainer, bool dofree=false)
  Add new graph heads.
- void share_children_of (SmacqGraph)
  Children of the speci ed graph will also become children of this.

## 6.21.2 Member Function Documentation

### 6.21.2.1 SmacqGraphContainer SmacqGraphContainer::clone (SmacqGraph newParent)

Recursively clone a graph.

The clone is made a child of newParent, unless newParent is NULL.

### 6.21.2.2 SmacqGraph SmacqGraphContainer::getInvariants (DTS , SmacqScheduler , DtsField &)

Return a subgraph containing only invariants over the speci ed eld.

The subgraph will contain only boolean lters that are applied to all objects in the graph (e.g. not within an OR) and that do NOT use the speci ed eld. The returned graph is newly allocated.

#### 6.21.2.3 std::string SmacqGraphContainer::print_query ()

Print the graph in re-parsable syntax.

So much for polymorphism

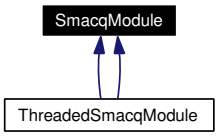
The documentation for this class was generated from the following  les:

- SmacqGraph.h
- libsmacq.cpp
- optimize.cpp
- SmacqGraph.cpp

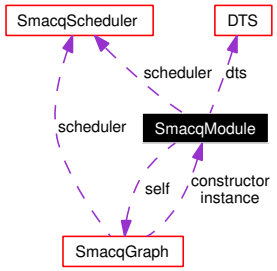
# 6.22 SmacqModule Class Reference

#include < SmacqModule-interface.h >

Inheritance diagram for SmacqModule:




Collaboration diagram for SmacqModule:




## 6.22.1 Detailed Description

A virtual base class for SMACQ modules.

This document describes the programming interface used by authors of data ow modules. These modules are dynamically loaded and may be instantiated multiple times. Global and static variables are therefore deprecated for most cases.

## Public Types

- typedefSmacqModule constructor_fr(structsmacq_init )

  SMACQ modules are object les that can be statically or dynamically loaded into a program.

## Public Member Functions

- SmacqModule(structsmacq_init context)

Most subclasses will de ne their own constructor which will initialize the instance based on the given context.

- virtual SmacqModule()

  A subclass can have its own destructor.

- virtual smacq_result consume(DtsObject datum, int &outchan)

  The consume() method is called when there is new data for a module to process.

- virtual smacq_result produce(DtsObject &datump, int &outchan)

  The produce() method is called when SMACQ expects an object to produce new data.

- virtual bool usesOtherFields(DtsField f)

  This method is called by the join optimizer.

## Protected Member Functions

- void comp_uses(dts_comparison c)
- dts_comparison parse_tests(int argc, char argv)

  Return a newly constructed dts_comparison datastructure from the given arguments.

- virtual DtsField uses eld(char name)

  This method wraps DTS::uses eld() but keeps track of what this module uses.

- void enqueue(DtsObject &, int outchan=0)

  Enqueue an object for output to the speci ed output channel.

## Protected Attributes

- UsesArray usesFields
- DTS dts

  Each module instance runs in the context of a DTS instance.

- SmacqScheduler scheduler

  Each module instance is run by a scheduler.

- SmacqGraph self

  A pointer to ourself in the current data ow graph.

## 6.22.2 Member Typedef Documentation

### 6.22.2.1 typedef SmacqModule SmacqModule::constructor_fn(struct smacq_init )

SMACQ modules are object files that can be statically or dynamically loaded into a program.

Each module should use the SMACQ_MODULE() macro to make sure that the module defines a constructor function that instantiates the class.

## 6.22.3 Member Function Documentation

### 6.22.3.1 smacq_result SmacqModule::consume(PDatsObject datum, int & outchan) [inline, virtual]

The consume() method is called when there is new data for a module to process.

It is passed a pointer to a data object and a reference to an output channel descriptor. The return code should be SMACQ_PASS if the object is not filtered out and SMACQ_FREE if it is. In addition, the return code can be OR'd with the following flags: SMACQ_ERROR specifies that there was a fatel error in the module. SMACQ_END signifies that the module wishes to never be called again.

Reimplemented in ThreadedSmacqModule and ThreadedSmacqModule

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

- SmacqModule-interface.h
- libsmacq.cpp
- SmacqModule.h
- parsing.cpp

## 6.23 SmacqModule::algebra Struct Reference

#include < SmacqModule-interface.h >

### 6.23.1 Detailed Description

The algebra element is optional and is used only by the data ow optimizer.

The following elements of the algebra structure are as follows: Vector speci es that the module can be used with a single input and a single output, or can be used with a vector of sets of arguments separated by semicolons and a corresponding vector of output channels. Boolean speci es that the module merely lters out some data and can be reordered in the data ow by an optimizer. Demux speci es that the module demultiplexes output data among multiple output chanels. If a demux module fails to set the demux bit, then the optimizer may produce disfunctional output.

### Public Attributes

- unsigned int **stateless**:1
- unsigned int **vector**:1
- unsigned int **annotation**:1
- unsigned int **demux**:1

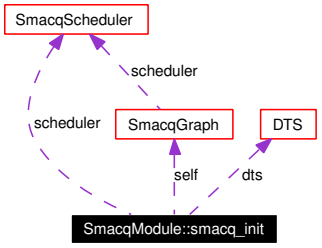
The documentation for this struct was generated from the following le:

- SmacqModule-interface.h

# 6.24 SmacqModule::smacq_init Struct Reference

#include < SmacqModule-interface.h >

Collaboration diagram for SmacqModule::smacq_init:




## 6.24.1 Detailed Description

This context structure is passed to SmacqModule constructors.

It will be destroyed after the constructor returns, but the elements it points to are guaranteed to be available during the lifetime of the object.

## Public Attributes

- SmacqScheduler scheduler
- bool is rst
- bool islast
- char argv
- int argc
- DTS dts
- SmacqGraph self

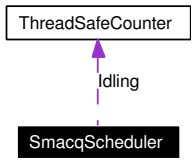
The documentation for this struct was generated from the following file:

- SmacqModule-interface.h

## 6.25 SmacqScheduler Class Reference

#include < SmacqScheduler.h >

Collaboration diagram for SmacqScheduler:




### 6.25.1 Detailed Description

This is a scheduler for processing any number of SmacqGraphContainer instances.

### Public Member Functions

- SmacqScheduler ()
  A default graph must be speci ed.
- void setDebug ()
  Set debug output.
- bool isDebug ()
  Get debug status.
- void seed_produce (SmacqGraphContainer)
  Cue the head(s) of the given graph to start producing data.
- void seed_produce (SmacqGraph startf)
  Cue the graph to start producing data.
- void input (SmacqGraphContainer g, DtsObject din)
  Queue an object for input to the speci ed graph.
- bool get (DtsObject &dout)
  Run until an output object is ready.
- smacq_result decide (SmacqGraph , DtsObject din)
  Process a single action or object.

- smacq_result decideContainer(SmacqGraphContainer, DtsObject din)

  Process a single action or object.

- bool busy_loop()

  Run to completion.

- void enqueue(SmacqGraph f, DtsObject d, int outchan)

  Handle an object produced by a currently running node.

- void queue_children(SmacqGraph_ptr, DtsObject d, int outchan)

  Handle an object produced by the speci ed node.

- bool element(DtsObject &dout)

  Process a single action or object.

- void start_threads(int numt)

  Create some threads to process the current workload.

## Public Attributes

- runq< SmacqGraph_ptr* consumeq
- runq< SmacqGraph_ptr* produceq
- runq< DtsObject> outputq

## Friends

- void iterative_scheduler_thread_start(void arg)

## 6.25.2 Constructor & Destructor Documentation

### 6.25.2.1 SmacqScheduler::SmacqScheduler () [inline]

A default graph must be speci ed.

Graph graph's init() method is called before anything else is done.

## 6.25.3 Member Function Documentation

### 6.25.3.1 bool SmacqScheduler::busy_loop ()

Run to completion.

Return false iff error.

### 6.25.3.2 smacq_result SmacqScheduler::decide (SmacqGraph g, DtsObject din)

Process a single action or object.

Return SMACQ_PASS or SMACQ_FREE.

### 6.25.3.3 smacq_result SmacqScheduler::decideContainer (SmacqGraphContainer g, DtsObject din)

Process a single action or object.

Return SMACQ_PASS or SMACQ_FREE.

### 6.25.3.4 bool SmacqScheduler::element (DtsObject & dout)

Process a single action or object.

dout will be set to NULL or to a returned object. Returns false if we're done and true if we want to be called again.

### 6.25.3.5 void SmacqScheduler::seed_producer (SmacqGraphContainer )

Cue the head(s) of the given graph to start producing data.

Otherwise data must be provided using the input() method.

### 6.25.3.6 void SmacqScheduler::start_threads (int num)

Create some threads to process the current workload.

They will exit when there is nothing to do.

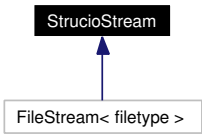
The documentation for this class was generated from the following files:

- SmacqScheduler.h
- libsmacq.cpp
- SmacqScheduler.cpp

# 6.26 StrucioStream Class Reference

#include < StrucioStream.h >

Inheritance diagram for StrucioStream:




## 6.26.1 Detailed Description

Pure-virtual base class for input streams.

## Public Member Functions

- StrucioStream (const char fname, const char fmode="rb")
- StrucioStream (const char fname, const int leno, const char fmode="rb")
  Construct from open le descriptor.
- void Follow ()
  Tell this stream to follow le changes.
- size_t Read (void ptr, size_t bytes)
  Read from stream.
- virtual size_t Write (void ptr, size_t bytes)=0
  Read from stream.
- DtsObject construct (DTS dts, dts_typeid t)
  Construct a xed-sized object.
- DtsObject construct (DTS dts, dts_typeid t, unsigned int size)
  Construct an object.

## Static Public Member Functions

- StrucioStream MagicOpen (const char lename, const char mode="rb")
  Return the appropriate subclass based on le type.

- StrucioStream MagicOpen(DtsObject fo)

  Open a le speci ed in a DtsObject and return a StrucioStream object for it.

## Protected Member Functions

- virtual size_t BasicRead(void ptr, size_t bytes)=0

  Read from stream.

- virtual bool Close()=0

  Close stream.

- virtual bool FdOpen ()=0
- bool Open()

  (Re)Open stream by name.

## Protected Attributes

- bool follow
- ino_t inode
- const char lename
- int fd

  We always keep a valid fd number around.

- const char mode

  Desired open mode (fopen syntax).

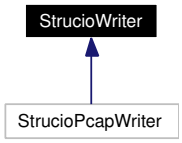
The documentation for this class was generated from the following le:

- StrucioStream.h

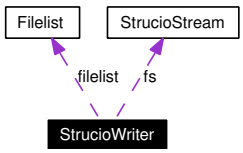
# 6.27 StrucioWriter Class Reference

#include < StrucioWriter.h >

Inheritance diagram for StrucioWriter:




Collaboration diagram for StrucioWriter:




## 6.27.1 Detailed Description

A le writer for structured data.

## Public Member Functions

- bool write (void record, size_t len)
- virtual void new le_hook ()
- void register_ lelist_stdin ()
- void register_ lelist_args (int argc, char argv)
- void register_ lelist_bounded (char index_location, long long lower, long long upper)
- void register_ le (char lename)
- void set_rotate_time(long long)
- void set_rotate_size(long long)
- void set_use_gzip(bool val)

## Protected Member Functions

- void newFilelist (Filelist )
- int openwrite ()

- void close_ le ()
- void format_ lename (char buf, size_t len)

## Protected Attributes

- char lename
- StrucioStream fs
- bool use_gzip
- long longoutputleft
- long longmax lesize
- long longmax leseconds
- time_t le_end_time
- Filelist lelist

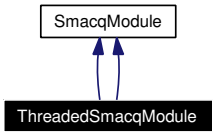
The documentation for this class was generated from the following le:

- StrucioWriter.h

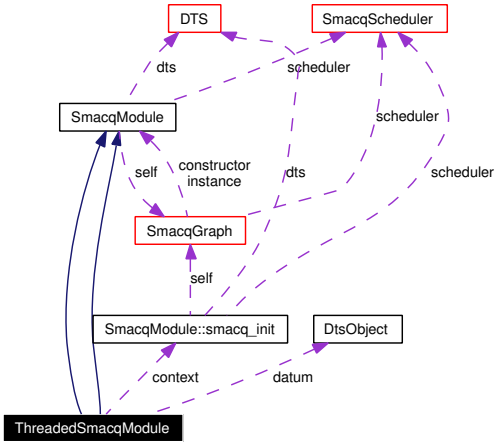
# 6.28 ThreadedSmacqModule Class Reference

#include < AsyncSmacqModule.h >

Inheritance diagram for ThreadedSmacqModule:




Collaboration diagram for ThreadedSmacqModule:




## 6.28.1 Detailed Description

A virtual base clase for SMACQ modules that are executed with in their own "thread" instead of being event-driven.

This is typically easier to program, but less ef cient than a regular SmacqModule. The only method that should be implemented by a subclass of ThreadedSmacqModule is thread().

## Public Member Functions

- smacq_result produce (DtsObject &, int &)
- smacq_result consume (DtsObject &, int &)

  The consume() method is called when there is new data for a module to process.

- ThreadedSmacqModule(smacq_init )
- smacq_result produce (DtsObject; int &)
- smacq_result consume(DtsObject; int &)

  The consume() method is called when there is new data for a module to process.

- ThreadedSmacqModule(smacq_init )

## Protected Member Functions

- virtual smacq_result thread(smacq_init context)=0

  This is the only method that subclasses should (and must) implement.

- virtual smacq_result thread(smacq_init context)=0

  This is the only method that subclasses should (and must) implement.

Methods used by the thread() implementation

- DtsObject smacq_read()

  Read a new data object to process.

- int smacq_ ush ()
- void smacq_decision(DtsObject datum, smacq_result result)

  Register a decision regarding an input object.

- void smacq_write(DtsObject datum, int outchan)

  Produce a new object.

Methods used by the thread() implementation

- DtsObject smacq_read()

  Read a new data object to process.

- int smacq_ ush ()
- void smacq_decision(DtsObject datum, smacq_result result)

  Register a decision regarding an input object.

- void smacq_write(DtsObject datum, int outchan)

  Produce a new object.

## Friends

- void run_thread (int args, ThreadedSmacqModule *ths)
- void run_thread (int args, ThreadedSmacqModule *ths)

## 6.28.2 Member Function Documentation

### 6.28.2.1 smacq_result ThreadedSmacqModule::consume (PktsObject, int &) [virtual]

The consume() method is called when there is new data for a module to process.

It is passed a pointer to a data object and a reference to an output channel descriptor. The return code should be SMACQ_PASS if the object is not filtered out and SMACQ_FREE if it is. In addition, the return code can be OR'd with the following flags: SMACQ_ERROR specifies that there was a fatel error in the module. SMACQ_END signifies that the module wishes to never be called again.

Reimplemented from SmacqModule

### 6.28.2.2 smacq_result ThreadedSmacqModule::consume (PktsObject, int &) [virtual]

The consume() method is called when there is new data for a module to process.

It is passed a pointer to a data object and a reference to an output channel descriptor. The return code should be SMACQ_PASS if the object is not filtered out and SMACQ_FREE if it is. In addition, the return code can be OR'd with the following flags: SMACQ_ERROR specifies that there was a fatel error in the module. SMACQ_END signifies that the module wishes to never be called again.

Reimplemented from SmacqModule

### 6.28.2.3 virtual smacq_result ThreadedSmacqModule::thread (smacq_init context) [protected, pure virtual]

This is the only method that subclasses should (and must) implement.

It performs all of the work of the module and uses the following methods to produce and consume data. This function should not return until the module is completely finished. Return SMACQ_END or SMACQ_ERROR.

6.28.2.4 virtual smacq_result ThreadedSmacqModule::thread (smacq_init context) [protected, pure virtual]

This is the only method that subclasses should (and must) implement.

It performs all of the work of the module and uses the following methods to produce and consume data. This function should not return until the module is completely nished. Return SMACQ_END or SMACQ_ERROR.

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

- AsyncSmacqModule.h
- ThreadedSmacqModule.h
- libsmacq.cpp
- ThreadedSmacqModule.cpp

# 6.29 ThreadSafeBoolean Class Reference

#include < ThreadSafe.h >

## 6.29.1 Detailed Description

A thread-safe boolean value.

## Public Member Functions

- bool get()
  Return the current status.
- bool set()
  Atempt to set the boolean to true. Return false iff already set.
- bool clear()
  Atempt to set the boolean to false. Return false iff already false.

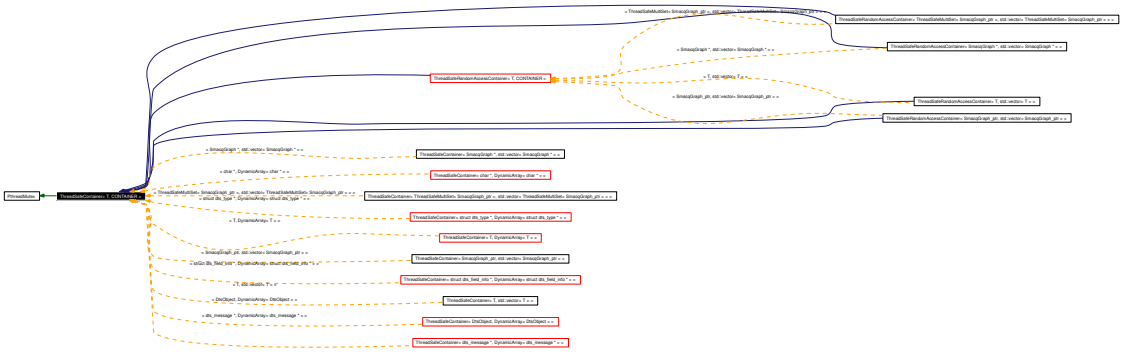
The documentation for this class was generated from the following le:

- ThreadSafe.h

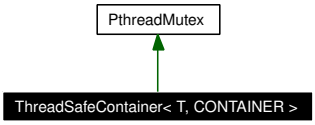
# 6.30 ThreadSafeContainer< T, CONTAINER > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeContainer< T, CONTAINER >:



Collaboration diagram for ThreadSafeContainer< T, CONTAINER >:




## 6.30.1 Detailed Description

template< typename T, typename CONTAINER > class ThreadSafeContainer< T, CONTAINER >

A base class for thread-safe containers.

## Public Types

- typedef CONTAINER::size_type sizetype

## Public Member Functions

- ThreadSafeContainer(int x)

- void clear ()
- CONTAINER::size_type size()
- bool empty ()
- CONTAINER snapshot()
- void resize(const typename CONTAINER::size_type x)
- template< class U> void foreach (U cb)
- template< typename U> CONTAINER::iterator ndIf (U cb)
- template< class U> bool has_if (U cb)
- template< typename U> bool erase_ rst_match (U cb)

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

- ThreadSafe.h

## 6.31 ThreadSafeCounter Class Reference

#include < ThreadSafe.h >

### 6.31.1 Detailed Description

An atomic, thread-safe counter.

### Public Member Functions

- gint increment ()
- gint decrement()
- gint get ()

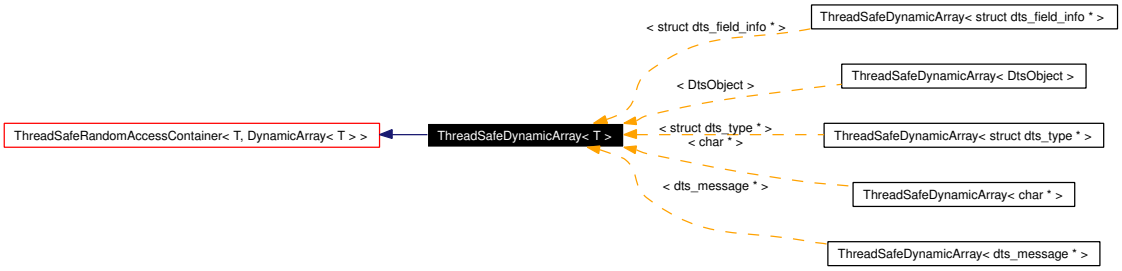
The documentation for this class was generated from the following le:

- ThreadSafe.h

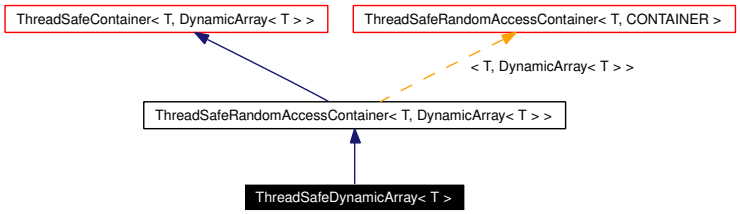
# 6.32 ThreadSafeDynamicArray< T > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeDynamicArray< T > :




Collaboration diagram for ThreadSafeDynamicArray< T > :




## 6.32.1 Detailed Description

template< typename T> class ThreadSafeDynamicArray< T >

A thread-safe dynamic array.

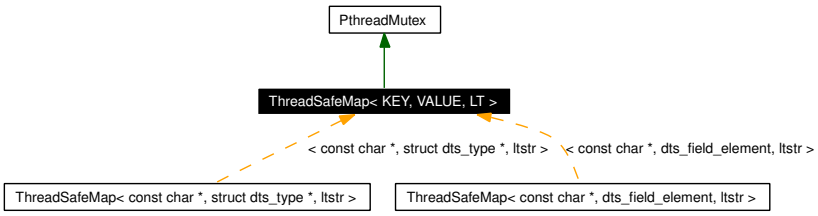
The documentation for this class was generated from the following le:

- ThreadSafe.h

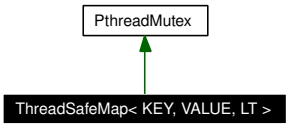
# 6.33 ThreadSafeMap< KEY, VALUE, LT > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeMap< KEY, VALUE, LT >:




Collaboration diagram for ThreadSafeMap< KEY, VALUE, LT >:




## 6.33.1 Detailed Description

template< typename KEY, typename VALUE, typename LT = std::less< KEY >>
class ThreadSafeMap< KEY, VALUE, LT >

A Thread-safe map based on an STL map.

## Public Member Functions

- bool get (KEY k, VALUE &v)
  Return true iff key already exists.
- VALUE & operator[] (KEY k)

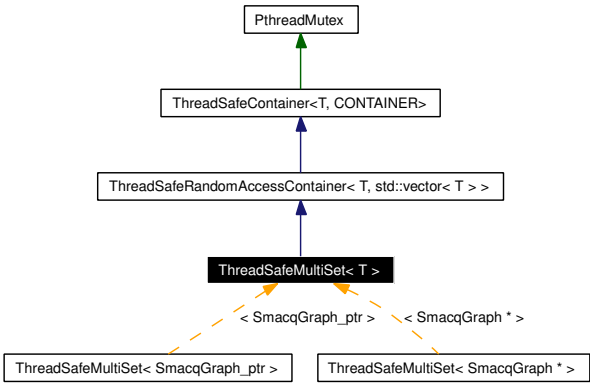
The documentation for this class was generated from the following file:

- ThreadSafe.h

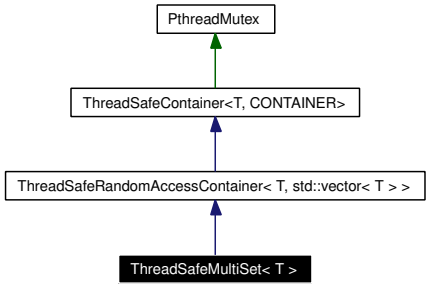
# 6.34 ThreadSafeMultiSet< T > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeMultiSet< T > :




Collaboration diagram for ThreadSafeMultiSet< T > :




## 6.34.1 Detailed Description

template< class T> class ThreadSafeMultiSet< T >

A thread-safe multiset.

## Public Member Functions

- void insert (const T &x)

- void erase(unsigned int i)
- bool erase(T x)

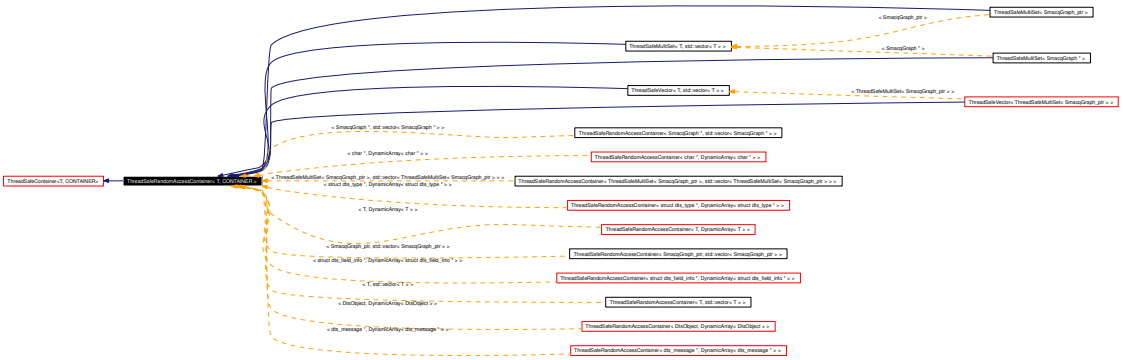
The documentation for this class was generated from the following file:

- ThreadSafe.h

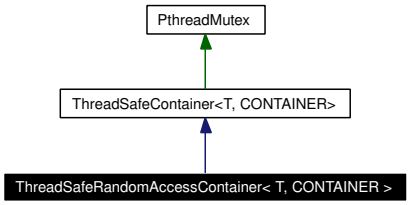
# 6.35 ThreadSafeRandomAccessContainer< T, CONTAINER > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeRandomAccessContainer< T, CONTAINER >:




Collaboration diagram for ThreadSafeRandomAccessContainer< T, CONTAINER >:




## 6.35.1 Detailed Description

template< typename T, typename CONTAINER > class ThreadSafeRandomAccessContainer< T, CONTAINER >

A base class for random-access thread-safe containers.

## Public Member Functions

- ThreadSafeRandomAccessContainer (int x)
- T & operator[] (size_t i)
- void push_back(T x)

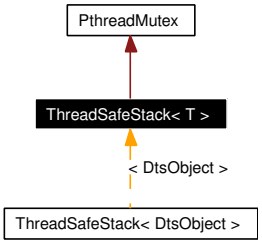
The documentation for this class was generated from the following file:

- ThreadSafe.h

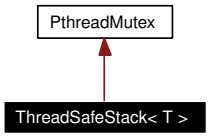
# 6.36 ThreadSafeStack< T > Class Template Reference

#include <ThreadSafe.h>

Inheritance diagram for ThreadSafeStack< T > :




Collaboration diagram for ThreadSafeStack< T > :




## 6.36.1 Detailed Description

template< class T> class ThreadSafeStack< T >

A thread-safe version of the STL stack class.

## Public Member Functions

- bool pop (T &ret)
- void push (T x)
- int size()

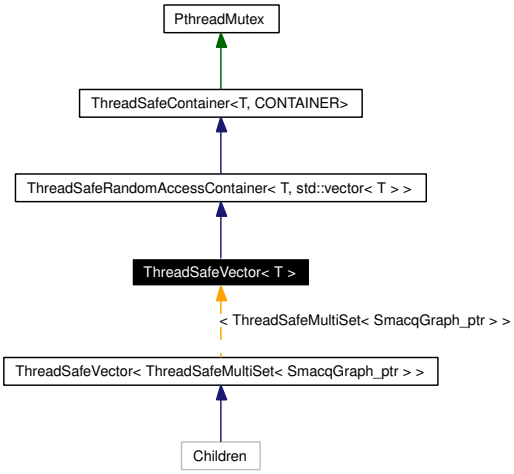
The documentation for this class was generated from the following file:

- ThreadSafe.h

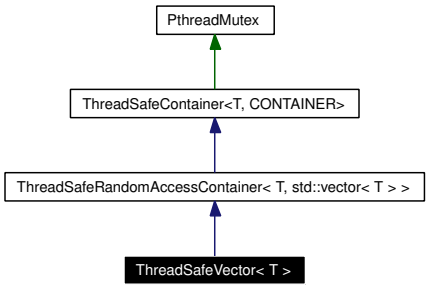
# 6.37 ThreadSafeVector< T > Class Template Reference

#include < ThreadSafe.h >

Inheritance diagram for ThreadSafeVector< T > :




Collaboration diagram for ThreadSafeVector< T > :




## 6.37.1 Detailed Description

template< typename T> class ThreadSafeVector< T >

A thread-safe version of the STL vector class.

## Public Member Functions

- ThreadSafeVector(int x)

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

- ThreadSafe.h

# Index


boost, 13
busy_loop
    SmacqScheduler, 60

clone
    SmacqGraph, 47
    SmacqGraphContainer, 51
construct
    DTS, 19
construct_fromstring
    DTS, 19
constructor_fn
    SmacqModule, 55
consume
    SmacqModule, 55
    ThreadedSmacqModule, 67

decide
    SmacqScheduler, 60
decideContainer
    SmacqScheduler, 60
do_shutdown
    SmacqGraph, 47
DTS, 15
    construct, 19
    construct_fromstring, 19
    DTS, 18
    freelist, 19
    requiretype, 19
    type_size, 19
    typenum_byname, 19
DtsField, 21
DtsObject, 22
DtsObject_, 23
DtsObject_
    dup, 25
    private_copy, 25
DtsObjectVec, 26
dup
    DtsObject_, 25
DynamicArray, 27

element
    SmacqScheduler, 60

FieldVec, 28
FieldVec
    has_unde ned, 29
    init, 29
FieldVecElement, 30
FieldVecHash, 31
FieldVecSet, 32
Filelist, 33
FilelistArgs, 34
FilelistBounded, 35
FilelistBounded
    next lename, 35
FilelistConstant, 37
FilelistError, 38
FilelistOneshot, 39
FilelistStdin, 40
freelist
    DTS, 19

getInvariants
    SmacqGraph, 48
    SmacqGraphContainer, 51

has_unde ned
    FieldVec, 29

init
    FieldVec, 29
    SmacqGraph, 48
intrusive_ptr_release

SmacqGraph, 48

move_children
SmacqGraph, 48

new_child
SmacqGraph, 48
next lename
FilelistBounded, 35

print_query
SmacqGraphContainer, 51
private_copy
DtsObject_, 25
PthreadMutex, 41

RecursiveLock, 43
requiretype
DTS, 19

seed_produce
SmacqScheduler, 60
SmacqGraph, 44
SmacqGraph
clone, 47
do_shutdown, 47
getInvariants, 48
init, 48
intrusive_ptr_release, 48
move_children, 48
new_child, 48
SmacqGraphContainer, 50
SmacqGraphContainer
clone, 51
getInvariants, 51
print_query, 51
SmacqModule, 53
SmacqModule
constructor_fn, 55
consume, 55
SmacqModule::algebra, 56
SmacqModule::smacq_init, 57
SmacqScheduler, 58
SmacqScheduler, 59
SmacqScheduler
busy_loop, 60
decide, 60
decideContainer, 60
element, 60
seed_produce, 60
SmacqScheduler, 59
start_threads, 60
start_threads
SmacqScheduler, 60
StrucioStream, 61
StrucioWriter, 63

thread
ThreadedSmacqModule, 67
ThreadedSmacqModule, 65
ThreadedSmacqModule
consume, 67
thread, 67
ThreadSafeBoolean, 69
ThreadSafeContainer, 70
ThreadSafeCounter, 72
ThreadSafeDynamicArray, 73
ThreadSafeMap, 74
ThreadSafeMultiSet, 75
ThreadSafeRandomAccessContainer, 77
ThreadSafeStack, 79
ThreadSafeVector, 80
type_size
DTS, 19
typenum_byname
DTS, 19