

SMACQ Reference Manual

Generated by Doxygen 1.3.5

Thu May 4 10:32:10 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	7
4.1	SMACQ Class List.	7
5	SMACQ Namespace Documentation	9
5.1	boost Namespace Reference.	9
6	SMACQ Class Documentation	11
6.1	DTS Class Reference	11
6.2	DtsField Class Reference.	15
6.3	DtsObject Class Reference.	16
6.4	DtsObject_ Class Reference	17
6.5	DtsObjectVec Class Reference.	20
6.6	DynamicArray< T > Class Template Reference	21

6.7	FieldVec Class Reference	22
6.8	FieldVecElement Class Reference	24
6.9	FieldVecHash T > Class Template Reference	25
6.10	FieldVecSet Class Reference	26
6.11	Filelist Class Reference	27
6.12	FilelistArgs Class Reference	28
6.13	FilelistBounded Class Reference	29
6.14	FilelistError Class Reference	30
6.15	FilelistOneshot Class Reference	31
6.16	FilelistStdin Class Reference	32
6.17	IterativeScheduler Class Reference	33
6.18	SmacqGraph Class Reference	35
6.19	SmacqGraphNode Class Reference	39
6.20	SmacqModule Class Reference	41
6.21	SmacqModule::algebra Struct Reference	44
6.22	SmacqModule::smacq_init Struct Reference	45
6.23	SmacqScheduler Class Reference	46
6.24	StrucioStream Class Reference	47
6.25	StrucioWriter Class Reference	49
6.26	ThreadedSmacqModule Class Reference	51

Chapter 1

System for Modular Analysis and Continuous Queries

Version:
2.4

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 OrsObject so that it compares the objects instead of the pointers to the objects)	9 .
---	---------------------

Chapter 3

SMACQ Hierarchical Index

3.1 SMACQ Class Hierarchy

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

DTS	11
DtsField	15
DtsObject	16
DtsObject_	17
DtsObjectVec	20
DynamicArray< T >	21
DynamicArray< bool >	21
FieldVec	22
FieldVecElement	24
FieldVecHask< T >	25
FieldVecSet	26
Filelist	27
FilelistArgs	28
FilelistBounded	29
FilelistError	30
FilelistOneshot	31
FilelistStdin	32
IterativeScheduler	33
SmacqGraphNode	39
SmacqGraph	35
SmacqModule	41
ThreadedSmacqModule	51
ThreadedSmacqModule	51
SmacqModule::algebra	44

SmacqModule::smacq_init	45
SmacqScheduler	46
StrucioStream	47
StrucioWriter	49

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. 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)	11
DtsField (DtsField is a vector class used to describe a field specification such as foo.bar.baz translated into numeric identifiers for fast lookup)	15
DtsObject (A DtsObject is an auto-pointer to a DtsObject_ instance)	16
DtsObject_ (DtsObject_ instances should only be used as DtsObject auto-pointers (the auto-pointer keeps track of reference counts for the user))	17
DtsObjectVec (A vector of DtsObject elements)	20
DynamicArray < T > (This is a wrapper around vector which grows the array as necessary to satisfy [] operations)	21
FieldVec (A vector of fields from a DtsObject)	22
FieldVecElement (An element of a FieldVec)	24
FieldVecHash < T > (A hash_map (table) for FieldVec)	25
FieldVecSet (A hash_set for FieldVec)	26
Filelist (A pure virtual base for classes that return lenames)	27
FilelistArgs (Return lenames from an argument vector)	28
FilelistBounded (Return lenames from an index le)	29
FilelistError (Never return a lename)	30
FilelistOneshot (Return a single lename)	31
FilelistStdin (Return lenames from STDIN)	32
IterativeScheduler (This is currently the only scheduler implementation)	33
SmacqGraph (A graph of SmacqGraphNode nodes)	35

SmacqGraphNode	(A node in a SmacqGraph Holds instance arguments, meta-data, etc)	39
SmacqModule	(A virtual base class for SMACQ modules)	41
SmacqModule::algebra	(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 channels. If a demux module fails to set the demux bit, then the optimizer may produce disfunctional output)	44
SmacqModule::smacq_init	(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)	45
SmacqScheduler	(SmacqScheduler is a typedef alias for IterativeScheduler)	46
StrucioStream	(Pure-virtual base class for input streams)	47
StrucioWriter	(A le writer for structured data)	49
ThreadedSmacqModule	(A virtual base class 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 this->read())	51

Chapter 5

SMACQ Namespace Documentation

5.1 boost Namespace Reference

5.1.1 Detailed Description

We overload the default == operator for [IntrusiveObject](#) so that it compares the objects instead of the pointers to the objects.

Functions

- `template< class T, class U> bool operator==(intrusive_ptr< T> const &a, intrusive_ptr< U> const &b)`
- `template< class T, class U> bool operator!=(intrusive_ptr< T> const &a, intrusive_ptr< U> const &b)`
- `template< class T> bool operator==(intrusive_ptr< T> const &a, T b)`
- `template< class T> bool operator!=(intrusive_ptr< T> const &a, T b)`
- `template< class T> bool operator==(T a, intrusive_ptr< T> const &b)`
- `template< class T> bool operator!=(T a, intrusive_ptr< T> const &b)`
- `template< class T> bool operator< (intrusive_ptr< T> const &a, intrusive_ptr< T> const &b)`
- `template< class T> void swap(intrusive_ptr< T> &lhs, intrusive_ptr< T> &rhs)`
- `template< class T> T get_pointer(intrusive_ptr< T> const &p)`
- `template< class T, class U> intrusive_ptr< T> static_pointer_cast(intrusive_ptr< U> const &p)`

- ~~template<~~ class T, class ~~U~~ intrusive_ptr< T > const_pointer_cast(intrusive_ptr< U > const &p)
- ~~template<~~ class T, class ~~U~~ intrusive_ptr< T > dynamic_pointer_cast(intrusive_ptr< U > const &p)
- ~~template<~~ class E, class T, class ~~Y~~std::basic_ostream<E, T> & operator<<(std::basic_ostream<E, T> &os, intrusive_ptr< Y > const &p)
- ~~template<~~> bool operator==< DtsObject_, DtsObject_> (const [DtsObject](#) &x, const [DtsObject](#) &y)

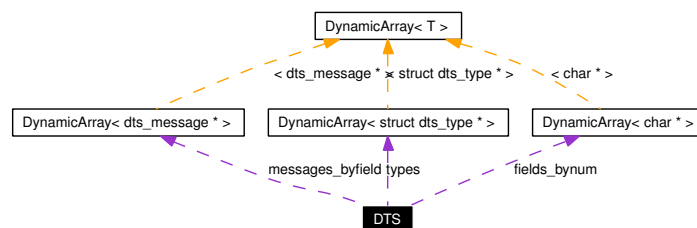
Chapter 6

SMACQ Class Documentation

6.1 DTS Class Reference

#include < dts.h >

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

- `int fromstring (dts_typeid, const char* data, DtsObject* data)`
- `int dts_lf (int type, void* p1, int len1, void* p2, int len2)`
- `int max_eld ()`
- `void use_master()`

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

Factory Methods

- `DtsObjectconstruct(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.
- `DtsObjectconstruct_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.
- `DtsObjectnewObject(dts_typeid)`
Return a new object of the given type.
- `DtsObjectnewObject(dts_typeid, int size)`
Return a new object of the given type and size.
- `DtsObjectreadObject (struct pickle* pickle, int fd)`

Field IDs

- `DtsFieldrequire_eld(char name)`
DtsObjects expose 0 or more `elds` (attributes) that can be accessed. Each `eld` is assigned a numeric identifier, `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. Convert the given `eld` name into a numeric identifier.
- `char _eld_getname(DtsField&f)`
Return the name of the specified `eld`.
- `char _eld_getname(dts_eld_element f)`
Return the name of the specified `eld`.

Type IDs

- `dts_typeidrequiretype(const char name)`

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. Load the specified type module (if it is not already loaded) and return the dynamically assigned numeric identifier for that type.

- `dts_typeid_t` `typename_bynum(const char name)`
If the specified type module is already loaded, this result is the same as `require_type()`. Unlike `require_type()` if the type is not loaded, -1 is returned.
- `char` `typename_bynum(const dts_typeid_t)`
Return the name of the given type.
- `dts_type_t` `type_bynum(const dts_typeid_t id)`
Return the type structure for the given type.
- `int` `type_size(const dts_typeid_t type)`
Return the size (in bytes) of the specified type. -1 if size is variable, -2 if type doesn't exist.

Interface to data testing system

- `int` `parsetest(dts_comparisoncomp, char test)`
- `int` `match(DtsObject datum, dts_comparisoncomps)`
- `dts_comparison` `parse_tests(int argc, char *argv)`
- `dts_operand` `parse_expr(int argc, char *argv)`

Messaging

- `void` `send_message(DtsObject dts_eld_element, dts_comparisoncomp)`
Send the given object to the given field of all objects that satisfy the comparison.
- `DtsObject` `msg_check(DtsObject, dts_eld_element_eld)`
Check for a message for this object and field.

Warnings

- `void` `set_no_warning()`
Don't warn.
- `bool` `warn_missing_elds()`
Get current setting.

Public Attributes

- `std::stack< DtsObject> freelist`

This freelist should only be used by `DtsObject` implementation. When an object is freed, it can be put on the freelist instead of being destroyed. `DtsObject::free()` will use objects on the freelist before constructing new objects.

Friends

- `int yysmacql_parse()`
- `int yydatalogparse()`
- `int yy_lterparse ()`
- `int yyexprparse()`

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

- `dts.h`
- `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

- 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.

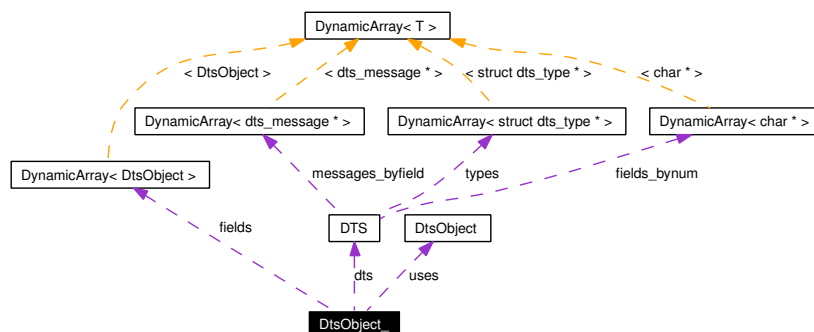
The documentation for this class was generated from the following file:

- DtsObject.h

6.4 DtsObject_ Class Reference

```
#include <DtsObject.h>
```

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).

Reference Counting

- 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`()
- std::vector< **DtsObject**> `get_all_elds`()
- 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_operandp)

Expr module uses this.

- DTS getDts() const
Pointer to the DTS used by this type.

Copy Constructors

- DtsObjectdup()
Return a new object with a copy of the data and a private eld vector. The eld vector is a copy of the original.
- DtsObjectprivate_copy()
Return a new object with shared data, but a private eld vector. The eld vector is a copy of the original.

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

- DtsObjectget_eld (DtsField& eldv, bool nowarn=false)
Return a eld object.
- DtsObjectget_eld (char s, bool nowarn=false)
Less efficient lookup by string.
- void attach_eld (DtsField& eld, DtsObject eld_data)

Friends

- [DtsObject](#)DTS::msg_check([DtsObject](#) dts_eld_element)

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

- DtsObject.h
- 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](#) [thash](#)(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.

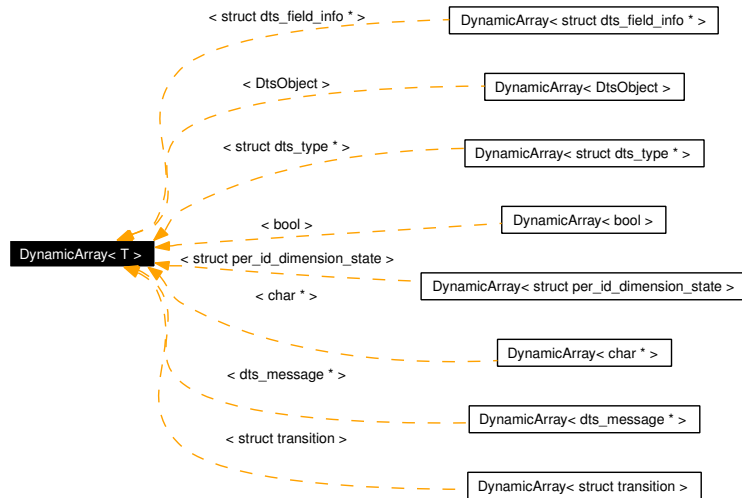
The documentation for this class was generated from the following file:

- [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)`

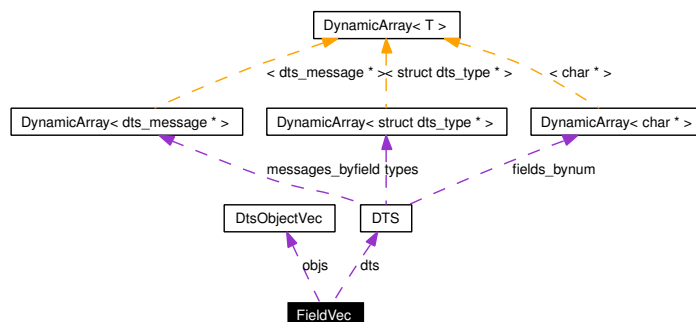
The documentation for this class was generated from the following file:

- `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 \(DtsObject\)](#)
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 [thisObject](#) Recomputed when [get elds\(\)](#) is called.
- [void init \(DTS , int argc, char argv\)](#)
Initialize eld vector from an argument vector. Deletes any previous contents.
- [FieldVec\(\)](#)
Construct an empty vector. Use [set\(\)](#) 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

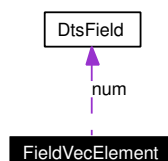
The documentation for this class was generated from the following le:

- 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

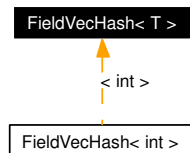
The documentation for this class was generated from the following file:

- 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](#)

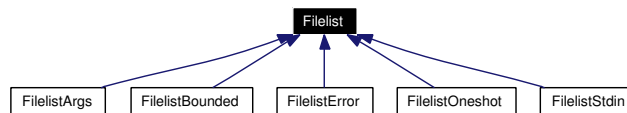
The documentation for this class was generated from the following file:

- 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 char next lename ()=0

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

- Filelist.h

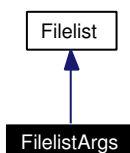
6.12 FilelistArgs Class Reference

```
#include <Filelist.h>
```

Inheritance diagram for FilelistArgs:



Collaboration diagram for FilelistArgs:



6.12.1 Detailed Description

Return file names from an argument vector.

Public Member Functions

- FilelistArgs (int, char *)
- char* next filename ()

Protected Attributes

- int strucio_argc
- char* strucio_argv

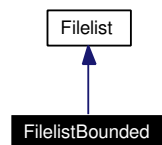
The documentation for this class was generated from the following file:

- Filelist.h

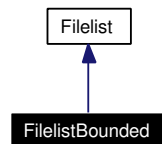
6.13 FilelistBounded Class Reference

```
#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)
- char next lename ()

Protected Attributes

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

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

- Filelist.h

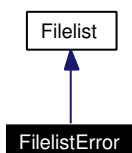
6.14 FilelistError Class Reference

```
#include <Filelist.h>
```

Inheritance diagram for FilelistError:



Collaboration diagram for FilelistError:



6.14.1 Detailed Description

Never return a file name.

Public Member Functions

- char* next filename ()

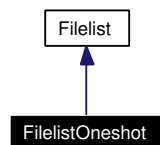
The documentation for this class was generated from the following file:

- Filelist.h

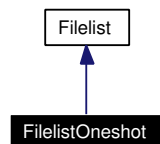
6.15 FilelistOneshot Class Reference

```
#include <Filelist.h>
```

Inheritance diagram for FilelistOneshot:



Collaboration diagram for FilelistOneshot:



6.15.1 Detailed Description

Return a single lename.

Public Member Functions

- FilelistOneshot(char lename)
- char next lename ()

Protected Attributes

- char le

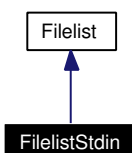
The documentation for this class was generated from the following le:

- Filelist.h

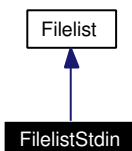
6.16 FilelistStdin Class Reference

```
#include <Filelist.h>
```

Inheritance diagram for FilelistStdin:



Collaboration diagram for FilelistStdin:



6.16.1 Detailed Description

Return le names from STDIN.

Public Member Functions

- char next lename ()

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

- Filelist.h

6.17 IterativeScheduler Class Reference

```
#include <IterativeScheduler-interface.h>
```

6.17.1 Detailed Description

This is currently the only scheduler implementation.

Public Member Functions

- [IterativeScheduler\(\)](#)
A default graph must be specified. Graph graph's init() method is called before anything else is done. Iff produce_ is.
- void [seed_produce\(SmacqGraphContainer\)](#)
Cue the head(s) of the given graph to start producing data. Otherwise data must be provided using the [input\(\)](#) method.
- void [seed_produce\(SmacqGraph startf\)](#)
Cue the graph to start producing data.
- void [input\(SmacqGraphContainer, DtsObjectdin\)](#)
Queue an object for input to the specified graph.
- smacq_result [get\(DtsObject&dout\)](#)
Run until an output object is ready.
- smacq_result [decide\(SmacqGraph, DtsObjectdin\)](#)
Process a single action or object.
- smacq_result [decide\(SmacqGraphContainer, DtsObjectdin\)](#)
Process a single action or object.
- bool [busy_loop\(\)](#)
Run to completion. Return false iff error.
- void [enqueue\(SmacqGraphf, DtsObjectd, int outchan\)](#)
Handle an object produced by a currently running node.
- void [queue_children\(SmacqGraphf, DtsObjectd, int outchan\)](#)
Handle an object produced by the specified node.

- `smacq_resultElement(DtsObject&dout)`
Process a single action or object.

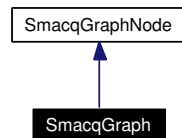
The documentation for this class was generated from the following files:

- `IterativeScheduler-interface.h`
- `IterativeScheduler.h`

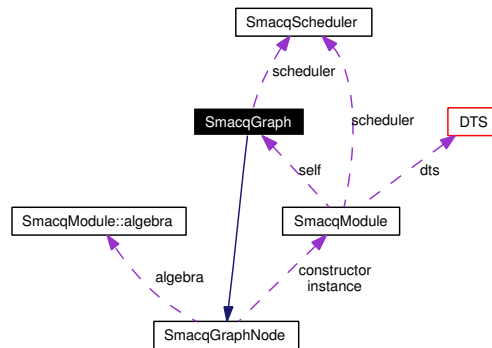
6.18 SmacqGraph Class Reference

```
#include <SmacqGraph-interface.h>
```

Inheritance diagram for SmacqGraph:



Collaboration diagram for SmacqGraph:



6.18.1 Detailed Description

A graph of **SmacqGraphNode** nodes.

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_inser**(**SmacqGraph**, **DTS**)

Insert a new graph between my parents and me.

- void [add_child](#)([SmacqGraph](#) child, unsigned int channel=0)
Add a new graph as one of my children.
- void [add_child](#)([SmacqGraphContainer](#) child, unsigned int channel=0)
Establish a parent/child relationship on the specified channel.
- void [remove_parent](#)([SmacqGraph](#) parent)
Remove the specified graph from the list of this graph's parents.
- void [remove_child](#)(int, int)
- void [remove_child](#)([SmacqGraph](#))
- 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< [PointerVector](#)< [SmacqGraph_ptr](#)> > [getChildren](#)() const
- 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. The new graph is automatically attached as a child of the current graph.
- [SmacqGraph clone](#)([SmacqGraph](#) newParent)
Recursively clone a graph. The clone is made a child of newParent, unless newParent is NULL.
- [SmacqGraphContainer newQuery](#)([DTS](#) , [SmacqScheduler](#), int argc, char* argv)
Parse a query and construct a new graph to execute it.

Public Member Functions

- `SmacqGraph(int argc, char* argv)`
- `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.
- `void print(FILE fh, int indent)`
Print the graph.
- `std::string print_query()`
Print the graph in re-parsable syntax.
- `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 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.
- `SmacqGraph getChildInvariants(DTS , SmacqScheduler, DtsField&)`
Same as `getInvariants()` but operates only on the graph's children.

Static Public Member Functions

- `void do_shutdown(SmacqGraph f)`
Shutdown a graph node (will propagate to parents and children). The node may be destroyed by this call.

Public Attributes

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

Friends

- void intrusive_ptr_add_ref ([SmacqGraph](#) o)
- void [intrusive_ptr_release](#)([SmacqGraph](#) o)
Decrement the reference count. If the refcount is 0, then clean-up references and destroy.

6.18.2 Member Function Documentation

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

If the module is vectorized, then add a new vector element; otherwise add it to channel 0

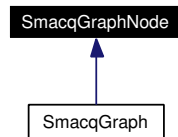
The documentation for this class was generated from the following les:

- [SmacqGraph-interface.h](#)
- [SmacqGraph.h](#)
- [optimize.cpp](#)
- [parsing.cpp](#)

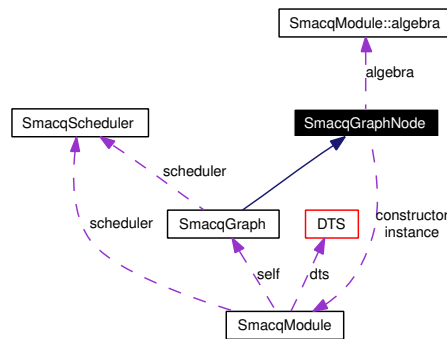
6.19 SmacqGraphNode Class Reference

```
#include < SmacqGraphNode.h >
```

Inheritance diagram for SmacqGraphNode:



Collaboration diagram for SmacqGraphNode:



6.19.1 Detailed Description

A node in a **SmacqGraph** holds instance arguments, meta-data, etc.

Public Member Functions

- void **init** (struct **SmacqModule::smacq_in8**)
Initialize a graph node's **set()** must be called rst.
- bool **set**(int argc, char* argv)
(Re-)Initialize module
- const int **getArgc**() const
Return argc.
- char* const **getArgv**() const

Return argv (do not modify).

Protected Attributes

- char argv
- int argc
- [SmacqModule::algebra](#) algebra
- bool shutdown
- bool mustProduce
- [SmacqModule](#) instance

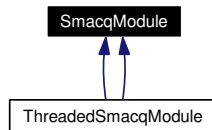
The documentation for this class was generated from the following file:

- SmacqGraphNode.h

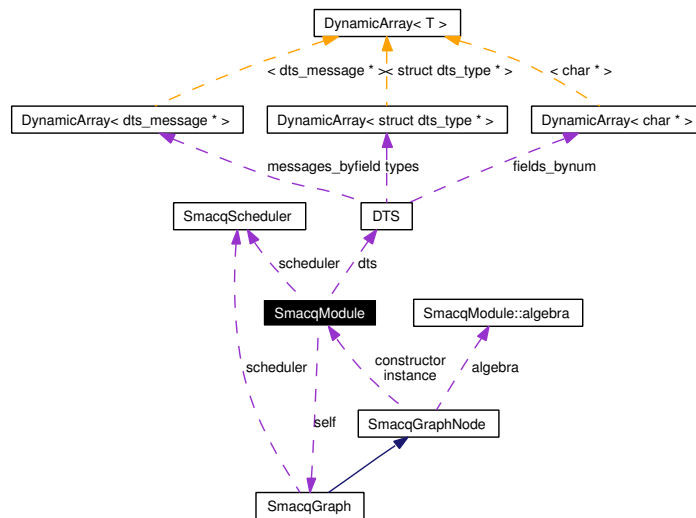
6.20 SmacqModule Class Reference

```
#include <SmacqModule-interface.h>
```

Inheritance diagram for SmacqModule:



Collaboration diagram for SmacqModule:



6.20.1 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

- `typedef SmacqModule constructor_fr(struct smacq_init)`

SMACQ modules are object les 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.

Public Member Functions

- `SmacqModule(struct smacq_init context)`

Most subclasses will define 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. 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 fatal error in the module. `SMACQ_END` signifies that the module wishes to never be called again.

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

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

- virtual `bool usesOtherField(DtsField f)`

This method is called by the join optimizer.

Protected Member Functions

- `void comp_uses(dts_comparisonc)`
- `dts_comparison parse_test(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 specified output channel.

Protected Attributes

- `UsesArrayusesFields`
- `DTS dts`
Each module instance runs in the context of a `DTS` instance.
- `SmacqSchedulerscheduler`
Each module instance is run by a scheduler.
- `SmacqGraph self`
A pointer to ourself in the current data ow graph.

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

- `SmacqModule-interface.h`
- `SmacqModule.h`
- `parsing.cpp`

6.21 SmacqModule::algebra Struct Reference

```
#include <SmacqModule-interface.h>
```

6.21.1 Detailed Description

The algebra element is optional and is used only by the data flow optimizer. The following elements of the algebra structure are as follows: Vector specifies 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 specifies that the module merely filters out some data and can be reordered in the data flow by an optimizer. Demux specifies that the module demultiplexes output data among multiple output channels. 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

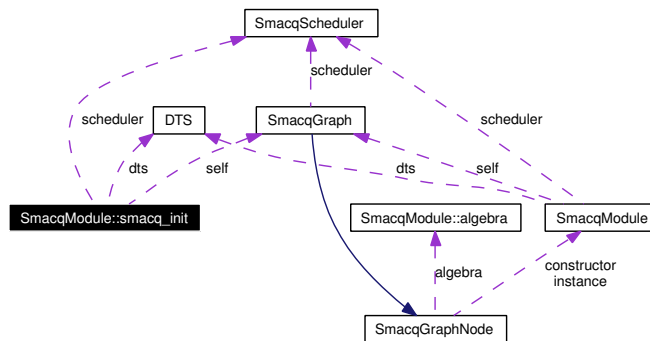
The documentation for this struct was generated from the following file:

- SmacqModule-interface.h

6.22 SmacqModule::smacq_init Struct Reference

```
#include <SmacqModule-interface.h>
```

Collaboration diagram for SmacqModule::smacq_init:



6.22.1 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 is last
- char argv
- int argc
- [DTS](#) dts
- [SmacqGraph](#) self

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

- SmacqModule-interface.h

6.23 SmacqScheduler Class Reference

```
#include < SmacqScheduler.h >
```

6.23.1 Detailed Description

SmacqScheduler is a typedef alias for [IterativeScheduler](#)

In the future, it may be a base class for multiple schedulers.

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

- SmacqScheduler.h

6.24 StrucioStream Class Reference

```
#include <StrucioStream.h>
```

6.24.1 Detailed Description

Pure-virtual base class for input streams.

Public Member Functions

- `StrucioStream(const char fname, const char mode="rb")`
- `StrucioStream(const char fname, const int leno, const char mode="rb")`
Construct from open file descriptor.
- `void Follow()`
Tell this stream to follow file changes.
- `size_t Read(void ptr, size_t bytes)`
Read from stream.
- `DtsObjectConstruct(DTS dts, dts_typeid t)`
Construct a fixed-sized object.
- `DtsObjectConstruct(DTS dts, dts_typeid t, unsigned int size)`
Construct an object.

Static Public Member Functions

- `StrucioStream MagicOpen(const char filename, const char mode="rb")`
Return the appropriate subclass based on file type.
- `StrucioStream MagicOpen(DtsObject fo)`
Open a file specified 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 `lname`
- int `fd`
We always keep a valid fd number around.
- const char `mode`
Desired open mode (fopen syntax).

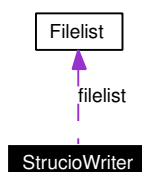
The documentation for this class was generated from the following files:

- `StrucioStream.h`

6.25 StrucioWriter Class Reference

```
#include <StrucioWriter.h>
```

Collaboration diagram for StrucioWriter:



6.25.1 Detailed Description

A le writer for structured data.

Public Member Functions

- `int write (void record, int 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(long long size)`
- `void set_use_gzip(bool val)`

Protected Member Functions

- `void newFilelist (Filelist)`
- `int openwrite ()`
- `void close_ le ()`

Protected Attributes

- `char lename`
- `FILE fh`
- `int use_gzip`
- `gzFilegzfh`

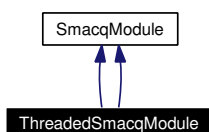
- long longoutputleft
- long longmax lesize
- int suf x
- [Filelist](#) lelist

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

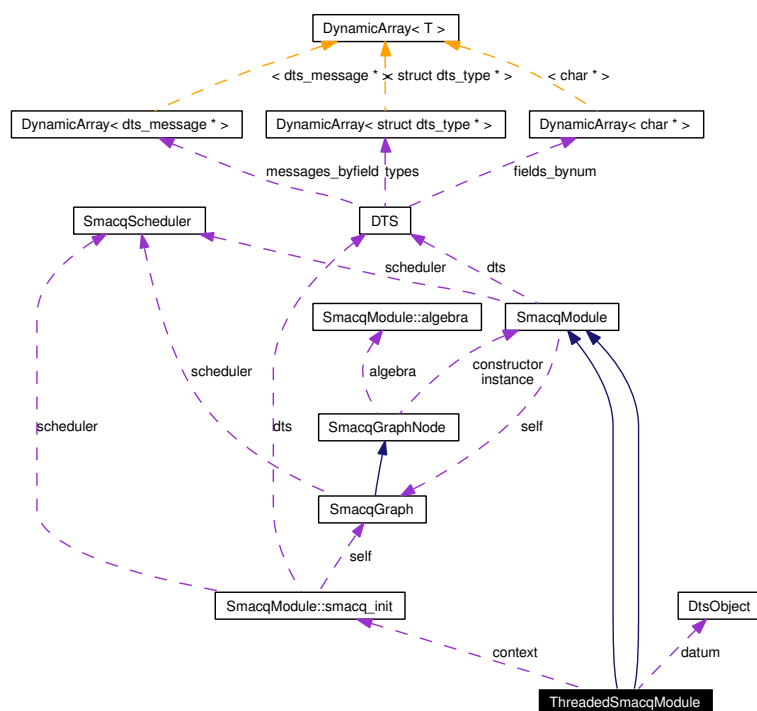
- StrucioWriter.h

```
#include <AsyncSmacqModule.h>
```

Inheritance diagram for ThreadedSmacqModule:



Collaboration diagram for ThreadedSmacqModule:



6.26.1 Detailed Description

A virtual base class for SMACQ modules that are executed with in their own "thread" instead of being event-driven. This is typically easier to program, but less efficient than a regular `SmacqModule`. The only method that should be implemented by a subclass of `ThreadedSmacqModule` is `isread()`.

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. 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 ltered out and SMACQ_FREE if it is. In addition, the return code can be OR'd with the following ags: SMACQ_ERROR speci es that there was a fatel error in the module. SMACQ_END signi es that the module wishes to never be called again.

- 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. 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 ltered out and SMACQ_FREE if it is. In addition, the return code can be OR'd with the following ags: SMACQ_ERROR speci es that there was a fatel error in the module. SMACQ_END signi es that the module wishes to never be called again.

- 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. 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.

- virtual smacq_result thread(smacq_init context)=0

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.

Methods used by the thread() implementation

- DtsObjectsmacq_read()
- Read a new data object to process.
- int smacq_ush()
- void smacq_decisio(DtsObjectdatum, 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

- [DtsObjectsmacq_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 \(int args, ThreadedSmacqModule ths\)](#)

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

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

Index

[boost](#),9

DTS, [11](#)

DtsField, [15](#)

DtsObject, [16](#)

DtsObject_, [17](#)

DtsObjectVec, [20](#)

DynamicArray, [21](#)

FieldVec, [22](#)

FieldVecElement, [24](#)

FieldVecHash, [25](#)

FieldVecSet, [26](#)

Filelist, [27](#)

FilelistArgs, [28](#)

FilelistBounded, [29](#)

FilelistError, [30](#)

FilelistOneshot, [31](#)

FilelistStdin, [32](#)

IterativeSchedule, [33](#)

move_children

[SmacqGraph](#), [38](#)

[SmacqGraph](#), [35](#)

[SmacqGraph](#)

 move_children, [38](#)

[SmacqGraphNode](#), [39](#)

[SmacqModule](#), [41](#)

[SmacqModule::algebra](#), [44](#)

[SmacqModule::smacq_init](#), [45](#)

[SmacqSchedule](#), [46](#)

[StrucioStream](#), [47](#)

[StrucioWriter](#), [49](#)

[ThreadedSmacqModule](#), [51](#)