SMACQ Reference Manual

Generated by Doxygen 1.4.7

Wed Jan 23 10:01:06 2008

Contents

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

You will need to instantiate a DTS type system object, a SmacqScheduler, and a SmacqGraph. Then use SmacqGraph::addQuery() to parse one or more queries. Finally use one of the SmacqScheduler methods like SmacqScheduler::input() or SmacqScheduler::busy_loop().

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 a DtsObject.

Chapter 2

SMACQ Hierarchical Index

2.1 SMACQ Class Hierarchy

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

Chapter 3

SMACQ Class Index

3.1 SMACQ Class List

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

Chapter 4

SMACQ Class Documentation

4.1 DTS Class Reference

`#include <dts.h>`

Inheritance diagram for DTS:Collaboration diagram for DTS:

4.1.1 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_field_element, dts_comparison *)

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

- DtsObject msg_check (DtsObject o, dts_field_element field)

  Check for a message for this obejct and field.

- void use_master ()

Make field 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 requirefield (char *name)
  Convert the given field name into a numeric identifier.
- char * field_getname (DtsField &f)
  Return the name of the specified field.
- std::string pyfield_getname (DtsField &f)
  A boost.python happy version.
- 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 require-type().
- 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 specified type.

Interface to data testing system

- int **parsetest** (dts_comparison *comp, char *test)
- int **match** (DtsObject datum, dts_comparison *comps)
- dts_comparison * **parse_tests** (std::string)
- dts_operand * **parse_expr** (std::string)

Warnings

- void set_no_warnings ()
  Don't warn.
- bool warn_missing_fields ()
  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 **yyfilterparse** ()
- int **yyexprparse** ()

4.1.2 Constructor & Destructor Documentation

4.1.2.1 DTS::DTS ()

Construct a new DTS.

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

4.1.3 Member Function Documentation

4.1.3.1 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.

4.1.3.2 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.

4.1.3.3 dts_typeid DTS::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.

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

4.1.3.4 int DTS::type_size (const dts_typeid *type*) [inline]

Return the size (in bytes) of the specified type.

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

4.1.3.5 int DTS::typenum_byname (const char * *name*) [inline]

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

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

4.1.4 Member Data Documentation

4.1.4.1 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. newObject() will use objects on the freelist before constructing new objects.

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

- dts.h
- libsmacq.cpp
- DTS.cpp
- parsing.cpp
- pickle.cpp

4.2 DtsDigest Class Reference

`#include <FieldVec.h>`

4.2.1 Detailed Description

A scalar hash of a DtsObjectVec.

Public Member Functions

- DtsDigest (const DtsDigest &d)
  Copy construct a hash.
- DtsDigest (FieldVec &v)
  Construct a hash from a FieldVec.
- DtsDigest (const DtsObjectVec &v)
  Construct a hash from a DtsObjectVec.
- DtsDigest (DtsObject &o)
  Construct a hash from a single DtsObject.
- bool **operator**< (const DtsDigest &y) const
- bool **operator==** (const DtsDigest &y) const

4.2.2 Constructor & Destructor Documentation

4.2.2.1 DtsDigest::DtsDigest (DtsObject & *o*) `[inline]`

Construct a hash from a single DtsObject.

XXX. this is wasteful and lame.

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

- FieldVec.h

4.3 DtsField Class Reference

`#include <DtsField.h>`

4.3.1 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

4.4 DtsObject Class Reference

`#include <DtsObject.h>`

4.4.1 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

4.5 DtsObject_ Class Reference

`#include <DtsObject.h>`

Inheritance diagram for DtsObject_:Collaboration diagram for DtsObject_:

4.5.1 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 initializaton (when it is assumed to only have a single user and therefore not require locking). Only the field cache is locked for thread safety.

A DtsObject is mainly accessed by requesting one of its fields with the getfield() method. Most objects will have a "string" and/or "double" field for accessing string or double representations of the object (if applicable).

To access the raw contents of an object, use the getdata() and getsize() methods or the dts_data_as(object, type) macro which will cast the data to the given type.

All DtsObjects have an underlying type that defines 0 or more fields that can be extracted from that object. The gettype() method returns the dts_typeid. However, a DtsObject may have additional fields added to it at runtime with the attach_field() method. To get a list of all of the current fields, you can use the fieldcache() method. To ensure that the fieldcache contains all available fields for the underlying type, precede the fieldcache() call with a call to prime_all_fields()

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_fields ()

  Instantiate all fields defined by the type of this object.

- void prime_field (dts_field_info *)

  Instantiate a specific field.

- std::vector< DtsObject > fieldcache ()

  Get a copy of all instantiated fields.

- int **write** (struct pickle *pickle, int fd)
- void **send** (dts_field_element fieldnum, dts_comparison *comparisons)
- void **send** (DtsField &field, 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 field vector.

- DtsObject private_copy ()

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

Meta-data Methods

- void **setsize** (int size)
- int **getsize** () const
- unsigned long getid () const

  Return the unique numeric identifier of the object.

- unsigned char * getdata () const

  Return the raw data contents of the object.

- std::string **pygetdata** () const
- dts_typeid gettype () const

  Return the type of the object.

- void **settype** (int type)

Initializers

- void **setdata** (void *data)
- void **setdatacopy** (const void *src)
- int **set_fromstring** (const char *datastr)

Field Access

- DtsObject getfield (DtsField &fieldv, bool nowarn=false)
  Return a field object.
- DtsObject getfield (char *s, bool nowarn=false)
  Less efficient lookup by string.
- void attach_field (DtsField &field, DtsObject field_data)
  Attach an object as a field.

Friends

- DtsObject **DTS::msg_check** (DtsObject, dts_field_element)

4.5.2 Member Function Documentation

4.5.2.1 DtsObject DtsObject_::dup ()

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

The field vector is a copy of the original.

4.5.2.2 std::vector<DtsObject> DtsObject_::fieldcache () [inline]

Get a copy of all instantiated fields.

You may want to use prime_all_fields() first.

4.5.2.3 DtsObject DtsObject_::private_copy ()

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

The field vector is a copy of the original.

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

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

4.6 DtsObjectVec Class Reference

`#include <FieldVec.h>`

4.6.1 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 fits this vector.

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

- FieldVec.h

4.7 DynamicArray< T > Class Template Reference

`#include <DynamicArray.h>`

Inheritance diagram for DynamicArray< T >:

4.7.1 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

4.8 FieldVec Class Reference

`#include <FieldVec.h>`

Collaboration diagram for FieldVec:

4.8.1 Detailed Description

A vector of fields from a DtsObject.

Public Member Functions

- bool getfields (DtsObject o)

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

- bool has_undefined () const

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

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

  Initialize field 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 field 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

4.8.2 Member Function Documentation

4.8.2.1 bool FieldVec::has_undefined () const [inline]

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

Recomputed when getfields() is called.

4.8.2.2 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

4.9 FieldVecDB< T > Class Template Reference

`#include <FieldVecDB.h>`

Inheritance diagram for FieldVecDB< T >:

4.9.1 Detailed Description

template<class T> class FieldVecDB< T >

A persistent store for per-vector state.

Public Member Functions

- FieldVecDB (const FieldVecDB< T > &d)

  Copy Constructor.

- FieldVecDB (const std::string &file)

  Instantiate a database with the given name (may include path).

- void **operator=** (const FieldVecDB &d)
- T **get** (const DtsObjectVec &v)
- void **put** (const DtsObjectVec &v, T &value)

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

- FieldVecDB.h

4.10 FieldVecDict< T > Class Template Reference

`#include <FieldVec.h>`

4.10.1 Detailed Description

template<class T> class FieldVecDict< T >

A dictionary for FieldVec keys.

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

- FieldVec.h

4.11 FieldVecElement Class Reference

`#include <FieldVec.h>`

Collaboration diagram for FieldVecElement:

4.11.1 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

4.12 FieldVecHash< T > Class Template Reference

`#include <FieldVec.h>`

Inheritance diagram for FieldVecHash< T >:

4.12.1 Detailed Description

template<class T> class FieldVecHash< T >

A map for FieldVec keys. The key itself is not stored, just a hash.

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

- FieldVec.h

4.13 FieldVecSet Class Reference

`#include <FieldVec.h>`

4.13.1 Detailed Description

A hash_set for FieldVec.

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

- FieldVec.h

4.14 Filelist Class Reference

`#include <Filelist.h>`

Inheritance diagram for Filelist:

4.14.1 Detailed Description

A pure virtual base for classes that return filenames.

Public Member Functions

- virtual void **nextfilename** (char *buf, int len)=0

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

- Filelist.h

4.15 FilelistArgs Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistArgs:Collaboration diagram for FilelistArgs:

4.15.1 Detailed Description

Return file names from an argument vector.

Public Member Functions

- **FilelistArgs** (int, char $**$)
- void **nextfilename** (char $*$, int)

Protected Attributes

- int **strucio_argc**
- char $**$ **strucio_argv**

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

- Filelist.h

4.16 FilelistBounded Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistBounded:Collaboration diagram for FilelistBounded:

4.16.1 Detailed Description

Return filenames from an index file.

Public Member Functions

- **FilelistBounded** (char *root, long long lower, long long upper)
- void nextfilename (char *, int)

Protected Attributes

- char * **indexfile**
- FILE * **index_fh**
- long long **lower_bound**
- long long **upper_bound**

4.16.2 Member Function Documentation

4.16.2.1 void FilelistBounded::nextfilename (char *, int) [inline, virtual]

Implements Filelist.

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

- Filelist.h

4.17 FilelistConstant Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistConstant:Collaboration diagram for FilelistConstant:

4.17.1 Detailed Description

Return a single filename.

Public Member Functions

- **FilelistConstant** (char *filename)
- void **nextfilename** (char *filename, int len)

Protected Attributes

- char * **file**

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

- Filelist.h

4.18 FilelistError Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistError:Collaboration diagram for FilelistError:

4.18.1 Detailed Description

Never return a file name.

Public Member Functions

- void **nextfilename** (char *buf, int len)

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

- Filelist.h

4.19 FilelistOneshot Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistOneshot:Collaboration diagram for FilelistOneshot:

4.19.1 Detailed Description

Return a single filename.

Public Member Functions

- **FilelistOneshot** (char ∗filename)
- void **nextfilename** (char ∗filename, int len)

Protected Attributes

- char ∗ **file**

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

- Filelist.h

4.20 FilelistStdin Class Reference

`#include <Filelist.h>`

Inheritance diagram for FilelistStdin:Collaboration diagram for FilelistStdin:

4.20.1 Detailed Description

Return file names from STDIN.

Public Member Functions

- void **nextfilename** (char *, int)

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

- Filelist.h

4.21 PthreadMutex Class Reference

`#include <ThreadSafe.h>`

Inheritance diagram for PthreadMutex:

4.21.1 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

4.22 RecursiveLock Class Reference

`#include <ThreadSafe.h>`

Collaboration diagram for RecursiveLock:

4.22.1 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

4.23 SmacqGraph Class Reference

`#include <SmacqGraph.h>`

4.23.1 Detailed Description

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

Public Member Functions

- SmacqGraph ()
  Default CTOR.
- SmacqGraph (ThreadSafeMultiSet< SmacqGraphNode_ptr > &children)
  Construct from a vector of Children.
- void addQuery (DTS *, SmacqScheduler *, std::string query)
  Parse a query and add it to a container.
- 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.
- bool empty ()
  Return true iff the container is empty.
- void print (FILE *fh, int indent)
  Print the graphs.
- std::string print_query ()
  Print the graph in re-parsable syntax.
- SmacqGraph * clone (SmacqGraphNode *newParent=NULL)
  Recursively clone a graph.
- void add_clone (SmacqGraphNode_ptr x, SmacqGraphNode *newParent)

Add a clone of a graph to this container.

- SmacqGraphNode * getInvariants (DTS *, SmacqScheduler *, DtsField &)
  Return a subgraph containing only invariants over the specified field.

- void optimize ()
  Preoptimize graph (unnecessary after init).

Combining Graphs

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

- void join (SmacqGraphNode *)
  Attach the specified graph onto the tail(s) of the graph(s).

- void join (SmacqGraph *, bool dofree=false)
  Attach the specified graph onto the tail(s) of the graph(s).

- void add_graph (SmacqGraphNode *)
  Add a new graph head.

- void add_graph (SmacqGraph *, bool dofree=false)
  Add new graph heads.

- void share_children_of (SmacqGraphNode *)
  Children of the specified graph will also become children of this.

4.23.2 Member Function Documentation

4.23.2.1 SmacqGraph * SmacqGraph::clone (SmacqGraphNode * *newParent* = NULL)

Recursively clone a graph.

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

4.23.2.2 SmacqGraphNode * SmacqGraph::getInvariants (DTS *, SmacqScheduler *, DtsField &)

Return a subgraph containing only invariants over the specified field.

The subgraph will contain only boolean 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.

4.23.2.3 std::string SmacqGraph::print_query ()

Print the graph in re-parsable syntax.

So much for polymorphism

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

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

4.24 SmacqGraphNode Class Reference

`#include <SmacqGraph.h>`

Inheritance diagram for SmacqGraphNode:Collaboration diagram for SmacqGraphNode:

4.24.1 Detailed Description

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

Parent/Child Relationships

- void join (SmacqGraphNode *)

  Attach the specified graph onto the tail(s) of the graph(s).

- void join (SmacqGraph *, bool dofree=false)

  Attach the specified graph onto the tail(s) of the graph(s).

- void replace (SmacqGraph *)

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

- void dynamic_insert (SmacqGraphNode *, DTS *)

  Insert a new graph between my parents and me.

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

  Add a new graph as one of my children.

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

  Establish a parent/child relationship on the specified channel.

- void remove_parent (SmacqGraphNode *parent)

  Remove the specified graph from the list of this graph's parents.

- void **remove_child_bynum** (int, int)
- void **remove_child** (SmacqGraphNode_ptr)
- void **replace_child** (int, int, SmacqGraphNode *newchild)
- void **replace_child** (int, int, SmacqGraph *newchild)
- void **replace_child** (SmacqGraphNode *oldchild, SmacqGraphNode *newchild)
- void **replace_child** (SmacqGraphNode *oldchild, SmacqGraph *newchild)
- void **remove_children** ()

- bool **live_children** ()
- bool **live_parents** ()
- const std::vector< ThreadSafeMultiSet< SmacqGraphNode_ptr > > **getChildren** ()
- static void move_children (SmacqGraphNode *from, SmacqGraphNode *to, bool addvector=false)

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).

- **SmacqGraphNode** (std::string)
- **SmacqGraphNode** (int argc, char **argv)
- SmacqGraphNode * 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.

Factories

- SmacqGraphNode * new_child (int argc, char **argv)

  Construct a new graph using the given arguments.

- SmacqGraphNode * clone (SmacqGraphNode *newParent)

  Recursively clone a graph.

Invariant Optimization

- SmacqGraphNode * getInvariants (DTS *, SmacqScheduler *, DtsField &)
  Return a subgraph containing only invariants over the specified field.
- SmacqGraphNode * 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

- static void do_shutdown (SmacqGraphNode_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** (SmacqGraphNode *o)
- void intrusive_ptr_release (SmacqGraphNode *o)

  Decrement the reference count.

4.24.2 Member Function Documentation

4.24.2.1 SmacqGraphNode * SmacqGraphNode::clone (SmacqGraphNode * *newParent*)

Recursively clone a graph.

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

4.24.2.2 void SmacqGraphNode::do_shutdown (SmacqGraphNode_ptr *f*) [static]

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

The node may be destroyed by this call.

4.24.2.3 SmacqGraphNode * SmacqGraphNode::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.

4.24.2.4 SmacqGraphNode * SmacqGraphNode::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.

4.24.2.5 void SmacqGraphNode::move_children (SmacqGraphNode * *from*, SmacqGraphNode * *to*, bool *addvector* = false) [inline, static]

4.24.2.6 SmacqGraphNode * SmacqGraphNode::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.

4.24.3 Friends And Related Function Documentation

4.24.3.1 void intrusive_ptr_release (SmacqGraphNode * *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
- SmacqGraph.cpp

4.25 SmacqModule Class Reference

`#include <SmacqModule-interface.h>`

Inheritance diagram for SmacqModule:Collaboration diagram for SmacqModule:

4.25.1 Detailed Description

A virtual base class for SMACQ modules.

This document describes the programming interface used by authors of dataflow 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_fn (struct smacq_init *)

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

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.

- 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 (std::string)

  Return a newly constructed dts_comparison datastructure from the given arguments.

- virtual DtsField usesfield (char *name)

  This method wraps DTS::usesfield() 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

- 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.

- SmacqGraphNode * self

  A pointer to ourself in the current dataflow graph.

Classes

- struct algebra

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

- struct smacq_init

  This context structure is passed to SmacqModule constructors.

- class **UsesArray**

4.25.2 Member Typedef Documentation

4.25.2.1 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.

4.25.3 Member Function Documentation

4.25.3.1 smacq_result SmacqModule::consume (DtsObject *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

4.26 SmacqModule::algebra Struct Reference

`#include <SmacqModule-interface.h>`

4.26.1 Detailed Description

The algebra element is optional and is used only by the dataflow 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 dataflow by an optimizer. Demux specifies 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 file:

- SmacqModule-interface.h

4.27 SmacqModule::smacq_init Struct Reference

`#include <SmacqModule-interface.h>`

Collaboration diagram for SmacqModule::smacq_init:

4.27.1 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 **isfirst**
- bool **islast**
- char ** **argv**
- int **argc**
- DTS * **dts**
- SmacqGraphNode * **self**

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

- SmacqModule-interface.h

4.28 SmacqScheduler Class Reference

`#include <SmacqScheduler.h>`

Collaboration diagram for SmacqScheduler:

4.28.1 Detailed Description

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

Public Member Functions

- SmacqScheduler ()
  A default graph must be specified.
- void setDebug ()
  Set debug output.
- bool isDebug ()
  Get debug status.
- void seed_produce (SmacqGraph *)
  Cue the head(s) of the given graph to start producing data.
- void seed_produce (SmacqGraphNode *startf)
  Cue the graph to start producing data.
- void input (SmacqGraph *g, DtsObject din)
  Queue an object for input to the specified graph.
- DtsObject get ()
  Run until an output object is ready, and return that object.
- smacq_result decide (SmacqGraphNode *, DtsObject din)
  Process a single action or object.
- smacq_result decideContainer (SmacqGraph *, DtsObject din)
  Process a single action or object.
- bool busy_loop ()
  Run to completion.

- void enqueue (SmacqGraphNode *f, DtsObject d, int outchan)

  Handle an object produced by a currently running node.

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

  Handle an object produced by the specified node.

- bool element (DtsObject &dout)

  Process a single action or object.

- DtsObject pyelement ()

  Created for the python smacq library Process a single action or object, if output is produced, return that output.

- bool done ()

  Return true if the query is done processing.

- void start_threads (int numt)

  Create some threads to process the current workload.

Public Attributes

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

4.28.2 Constructor & Destructor Documentation

4.28.2.1 SmacqScheduler::SmacqScheduler () [inline]

A default graph must be specified.

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

4.28.3 Member Function Documentation

4.28.3.1 bool SmacqScheduler::busy_loop ()

Run to completion.

Return false iff error.

4.28.3.2 smacq_result SmacqScheduler::decide (SmacqGraphNode * *g*, DtsObject *din*)

Process a single action or object.

Return SMACQ_PASS or SMACQ_FREE.

4.28.3.3 smacq_result SmacqScheduler::decideContainer (SmacqGraph * *g*, DtsObject *din*)

Process a single action or object.

Return SMACQ_PASS or SMACQ_FREE.

4.28.3.4 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.

4.28.3.5 DtsObject SmacqScheduler::get ()

Run until an output object is ready, and return that object.

Return NULL if execution completes without producing object.

4.28.3.6 void SmacqScheduler::seed_produce (SmacqGraph *)

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

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

4.28.3.7 void SmacqScheduler::start_threads (int *numt*)

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

4.29 StrucioStream Class Reference

`#include <StrucioStream.h>`

Inheritance diagram for StrucioStream:

4.29.1 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 fileno, const char *fmode="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.

- virtual size_t Write (void *ptr, size_t bytes)=0

  Read from stream.

- DtsObject construct (DTS *dts, dts_typeid t)

  Construct a fixed-sized object.

- DtsObject construct (DTS *dts, dts_typeid t, unsigned int size)

  Construct an object.

Static Public Member Functions

- static StrucioStream * MagicOpen (const char *filename, const char *mode="rb")

  Return the appropriate subclass based on file type.

- static 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 * **filename**
- 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 file:

- StrucioStream.h

4.30 StrucioWriter Class Reference

`#include <StrucioWriter.h>`

Inheritance diagram for StrucioWriter:Collaboration diagram for StrucioWriter:

4.30.1 Detailed Description

A file writer for structured data.

Public Member Functions

- bool **write** (void *record, size_t len)
- virtual void **newfile_hook** ()
- void **register_filelist_stdin** ()
- void **register_filelist_args** (int argc, char **argv)
- void **register_filelist_bounded** (char *index_location, long long lower, long long upper)
- void **register_file** (char *filename)
- 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_file** ()
- void **format_filename** (char *buf, size_t len)

Protected Attributes

- char * **filename**
- StrucioStream * **fs**
- bool **use_gzip**
- long long **outputleft**
- long long **maxfilesize**
- long long **maxfileseconds**
- time_t **file_end_time**
- Filelist * **filelist**

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

- StrucioWriter.h

4.31 ThreadedSmacqModule Class Reference

`#include <ThreadedSmacqModule.h>`

Inheritance diagram for ThreadedSmacqModule:Collaboration diagram for ThreadedSmacqModule:

4.31.1 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 efficient 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 *)

Static Public Member Functions

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

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_flush** ()
- 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_flush** ()
- 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)

4.31.2 Member Function Documentation

4.31.2.1 smacq_result ThreadedSmacqModule::consume (DtsObject, 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.

4.31.2.2 smacq_result ThreadedSmacqModule::consume (DtsObject, 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.

4.31.2.3 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.

4.31.2.4 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 finished. Return SMACQ_END or SMACQ_ERROR.

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

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

4.32 ThreadSafeBoolean Class Reference

`#include <ThreadSafe.h>`

4.32.1 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 file:

- ThreadSafe.h

4.33 ThreadSafeContainer< T, CONTAINER > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeContainer< T, CONTAINER >:Collaboration diagram for ThreadSafeContainer< T, CONTAINER >:

4.33.1 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 **findIf** (U cb)
- template<class U> bool **has_if** (U cb)
- template<typename U> bool **erase_first_match** (U cb)

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

- ThreadSafe.h

4.34 ThreadSafeCounter Class Reference

`#include <ThreadSafe.h>`

4.34.1 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 file:

- ThreadSafe.h

4.35 ThreadSafeDynamicArray< T > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeDynamicArray< T >:Collaboration diagram for ThreadSafeDynamicArray< T >:

4.35.1 Detailed Description

template<typename T> class ThreadSafeDynamicArray< T >

A thread-safe dynamic array.

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

- ThreadSafe.h

4.36 ThreadSafeMap< KEY, VALUE, LT > Class Template Reference

`#include <ThreadSafe.h>`

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

4.36.1 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

4.37 ThreadSafeMultiSet< T > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeMultiSet< T >:Collaboration diagram for ThreadSafeMultiSet< T >:

4.37.1 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

4.38 ThreadSafeRandomAccessContainer< T, CONTAINER > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeRandomAccessContainer< T, CONTAINER >:Collaboration diagram for ThreadSafeRandomAccessContainer< T, CONTAINER >:

4.38.1 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

4.39 ThreadSafeStack< T > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeStack< T >:Collaboration diagram for ThreadSafeStack< T >:

4.39.1 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

4.40 ThreadSafeVector< T > Class Template Reference

`#include <ThreadSafe.h>`

Inheritance diagram for ThreadSafeVector< T >:Collaboration diagram for ThreadSafeVector< T >:

4.40.1 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