SASSY

0.0

Generated by Doxygen 1.8.5

Thu Nov 2 2017 18:12:03

Contents

Chapter 1

SASSY

SASSY, the Software Architecture Support System, aims to make the power of knowledge databases available to the process of developing a software architecture.

The SASSY project has several sub-projects which are likely to be useful in their own right. These API pages document the libraries of these sub-projects.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 rdf Namespace Reference

The namespace for the Resource Description Framework interface.

Classes

- class vxt

  An exception object with severity levels.
- class URI

  A shared pointer with constructors for the URI_ class.
- class Model

  A shared pointer with constructors for the Model_ class.
- class ResourceNode

  A shared pointer with constructors for the ResourceNode_ class.
- class LiteralNode

  A shared pointer with constructors for the LiteralNode_ class.
- class BlankNode

  A shared pointer with constructors for the BlankNode_ class.
- class Statement

  A shared pointer with constructors for the Statement_ class.
- class Parser

  A shared pointer with constructors for the Parser_ class.
- class Serializer

  A shared pointer with constructors for the Serializer_ class.
- class Stream

  A shared pointer with constructors for the Stream_ class.
- class Query

  A shared pointer with constructors for the Query_ class.
- class ErrorClient

  Client that is notified of errors and/or warnings.
- class Universe

  A singleton class responsible for managing the World objects.
- struct Format

  Instructions for converting a node to a string.
- class Prefixes

  Manages the prefixes and namespaces for a World.

- class Literal

  Hold the value, language and data type for an RDF literal.
- class World_

  An abstract class defining the methods for an RDF World.
- class Model_

  An abstract class defining the methods for an RDF Model.
- class Node_

  An abstract class defining the methods for an RDF Node.
- class ResourceNode_

  An abstract class defining the methods for an RDF Resource Node.
- class LiteralNode_

  An abstract class defining the methods for an RDF Literal Node.
- class BlankNode_

  An abstract class defining the methods for an RDF blank Node.
- class Parser_

  An abstract class defining the methods for an RDF Parser.
- class Query_

  An abstract class defining the methods for an RDF Query.
- class QueryResult_

  An abstract class defining the methods for an RDF Query Result.
- class QueryResults_

  An abstract class defining the methods for a set of Query Results.
- class QueryString

  A class for assistingin the preparation of a SPARQL query.
- class Serializer_

  An abstract class defining the methods for an RDF Serializer.
- class Statement_

  An abstract class defining the methods for an RDF Statement.
- class Stream_

  An abstract class defining the methods for an RDF Stream.
- class URI_

  An abstract class defining the methods for an RDF URI.

Typedefs

- using World = std::shared_ptr< World_ >

  A shared pointer to a World object.
- using WorldRef = std::weak_ptr< World_ >

  A weak shared pointer to a World object.
- using NodeRef = std::weak_ptr< Node_ >

  A weak shared pointer to a Node object.
- using QueryResults = std::shared_ptr< QueryResults_ >

  A shared pointer to a set of query results.
- using StatementRef = std::weak_ptr< Statement_ >

  A weak shared pointer to a statement.
- using Node = std::shared_ptr< Node_ >

  A shared pointer to a Node object.

Enumerations

- enum Severity {
  Emergency, Alert, Critical, Error,
  Code, Warning, Notice, Info,
  Debug }

  Severity levels for log messages.
- enum Concept {
  Container, **Bag**, **Sequence**, **Alternative**,
  aboutEach, **List**, **first**, **rest**,
  nil, **Statement**, **object**, **predicate**,
  subject, **Resource**, **Class**, **subClassOf**,
  type, **Property**, **subPropertyOf**, **domain**,
  range, **ConstraintProperty**, **ConstraintResource**, **Description**,
  label, **seeAlso**, **comment**, **isDefinedBy** }

  An enumeration of RDF and RDFS concepts.
- enum DataType {
  UNDEF, **PlainLiteral**, **XMLLiteral**, **XHTML**,
  String, **Boolean**, **Decimal**, **Integer**,
  Double, **Float**, **Data**, **Time**,
  DateTime, **DateTimeStamp**, **Year**, **Month**,
  Day, **YearMonth**, **MonthDay**, **Duration**,
  YearMonthDuration, **DayTimeDuration**, **Byte**, **Short**,
  Int, **Long**, **UnsignedByte**, **UnsignedShort**,
  UnsignedLong, **PositiveInteger**, **NonNegativeInteger**, **NegativeInteger**,
  NonPositiveInteger, **HexBinary**, **Base64Binary**, **AnyURI**,
  Language, **NormalizedString**, **Token**, **NMTOKEN**,
  Name, **NCName** }

  A enumeration of the data types available for RDF literals.

Functions

- bool operator== (Statement, Statement)

  Check for equality of two statements.
- bool operator== (URI, URI)

  Check for equality of two URIs.
- template<class C , class P , typename T >
  T * deref (std::shared_ptr< C > a)

  A template function that converts a shared pointer into the corresponding librdf pointer.

6.1.1 Detailed Description

The namespace for the Resource Description Framework interface.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum rdf::Severity

Severity levels for log messages.

Enumerator

Emergency A fault has been detected which may compromise the computer.

Alert A configuration error has been detected.

Critical The program cannot continue and may have corrupted its data.

Error The program cannot continue.

Code A programming error has been detected.

Warning There is a problem but the program can continue.

Notice Something is odd.

Info Information messages.

Debug Debugging messages.

6.2 SASSY Namespace Reference

The namespace for the Software Architecture Support System project.

Namespaces

- cdi

  The namespace for the Common Development Infrastructure.
- cfi

  The namespace for the Common Facilities Infrastructure components.

Classes

- class Path

  Manipulate path strings.
- class sxt

  An exception object with severity levels.

Typedefs

- typedef void(∗ AsyncTestEventFn)(const std::string &s)

  Pointer to function used to enqueue a test event.

Enumerations

- enum Severity {
  Emergency, Alert, Critical, Error,
  Code, Warning, Notice, Info,
  Debug }

  Severity levels for log messages.

Functions

- void trim (std::string &s)

  rip off leading and trailing white spaces
- int split (csr text, std::vector< std::string > &result)

  split a string into sub-strings at spaces
- std::string expandMacros (csr s)

  expand a string containing $ macros
- void defaultAsyncTestEvent (const std::string &s)

  Function used to enqueue a test event.

Variables

- AsyncTestEventFn asyncTestEvent = SASSY::defaultAsyncTestEvent

  Pointer to function used to enqueue a test event.

6.2.1 Detailed Description

The namespace for the Software Architecture Support System project. The namespace for the project.

6.2.2 Typedef Documentation

6.2.2.1 typedef void(∗ SASSY::AsyncTestEventFn)(const std::string &s)

Pointer to function used to enqueue a test event.

Parameters

s	Identity of the event.

6.2.3 Enumeration Type Documentation

6.2.3.1 enum SASSY::Severity

Severity levels for log messages.

Enumerator

Emergency A fault has been detected which may compromise the computer.

Alert A configuration error has been detected.

Critical The program cannot continue and may have corrupted its data.

Error The program cannot continue.

Code A programming error has been detected.

Warning There is a problem but the program can continue.

Notice Something is odd.

Info Information messages.

Debug Debugging messages.

6.2.4 Function Documentation

6.2.4.1 void SASSY::defaultAsyncTestEvent (const std::string & *s*)

Function used to enqueue a test event.

The default function is defined in cfi/log.cpp and soes nothing

Parameters

s	Identity of the event.

6.2.4.2 string SASSY::expandMacros (csr *s*)

expand a string containing $ macros

Parameters

s	The string to expand.

Returns

A string with $ macros replaced.

6.2.4.3 int SASSY::split (csr *text*, std::vector< std::string > & *result*)

split a string into sub-strings at spaces

Parameters

text	the string to split
result	the vector of sub strings

Returns

a number of sub-strings

6.2.4.4 void SASSY::trim (std::string & *s*)

rip off leading and trailing white spaces

Parameters

s	the string to trim

6.3 SASSY::cdi Namespace Reference

The namespace for the Common Development Infrastructure.

Classes

- class Tester
  Responsible for managing the testing.
- class test_exception
  throw this to abandon a particular test case
- class scenario_exception
  throw this to abandon an entire scenario
- class TestEvent
  Represent an event in the object under test.
- class TestEventQueue
  Responsible for queuing TestEvents.
- class TestFactory
  Responsible for creating an object that manages the resources for the entire test.
- class TestFT
  Responsible for creating a test object of the required type.
- class AbstractTest
  Responsible for managing resources needed for the entire test.
- class TestT

Responsible for installing a factory for making test objects.

- class DefaultTest

  Provide a default version of the AbstractTest object.
- class ScenarioFactory

  Define a type for scenario factories.
- class ScenarioFT

  Responsible for creating a scenario of some type.
- struct ScenarioResults

  Responsible for storing the results of testing for a scenario.
- class AbstractScenario

  Define a type for scenarios.
- class ScenarioT

  Responsible for installing a factory to create scenarios of the required type.
- class DefaultScenario

  Provide a default scenario object.
- class TestCaseFactory

  Define a base type for test case factories.
- class TestCaseFT

  Responsible for creating a test case of some type.
- class AbstractTestCase

  Base class for test cases.
- class TestCaseT

  Responsible for installing a factory for the test case.
- class AsyncTestCase

  Responsible for handling asynchronous tests.
- class AsyncTestCaseT

  Responsible for installing the factory for the test case type.
- class CallTrace

  Class to create a call trace.
- class NullTrace

  A class for dummy trace objects.
- class Trace

  This class is used to create trace log entries.
- class Tracer

  This class manages the tracing for an application.

6.3.1 Detailed Description

The namespace for the Common Development Infrastructure. The cdi namespace is for components that are used during development of SASSY but are not part of the product used by a normal user.

6.4 SASSY::cfi Namespace Reference

The namespace for the Common Facilities Infrastructure components.

Classes

- struct DiscoverPointer

  Holds a pointer to a discoverable object.
- class Discoverable

  A mixin class that makes its owner discoverable.
- class DiscoveryMgr

  Manager for discoverable objects.
- class FD

  Wrap file descriptors in a class to ensure closed.
- class fdoutbuf

  An output stream buffer.
- class fdostream

  A file descriptor output stream.
- class fdinbuf

  An input stream buffer.
- class fdistream

  A file descriptor input stream.
- class Semaphore

  A semaphore for managing exclusive access to resources across multiple processes.
- class SemaphoreLock

  Mutual exclusion lock.
- class logstream

  A stream class specialized for logging.
- class logger

  An abstract base class used by the log stream to write logs.
- class FormattingLogger

  Class for producing a formatted log message.
- class StdLogger

  Class for logging to the std output streams.
- class FileLogger

  Class for logging to a file.
- class PlainFileLogger
- class UDPLogger

  Class for logging to a logging server.
- class TraceLogger

  Class for logging tracing output.
- class SystemLogger

  logger that interfaces to the Linux syslog
- class PlugIn

  Base class for objects loaded from dynamically loaded libraries.
- class AutoRunPlugIn

  Base class for plug ins that are run immediately that the library is loaded.
- class PlugInFactory

  Base class for factories that create plug-in objects.
- class PlugInFamilyFactoryT

  Template base class for families of plug-in factories.
- class PlugInFactoryT

  Template class for factories.
- struct PlugInDescriptor

  Descriptor for plug-ins that can be used by the application.

- struct PlugInDetails

  Details of plug-ins as provided by the loaded library.
- class PlugInLib

  Manages an instance of a dynamically loaded shared library.
- class PlugInMgr

  Manage the handling of plug-in shared libraries.
- class AbstractChildProcess

  Abstract child process interface for ChildProcessMgr.
- class ChildProcess

  Represents the state of a child process.
- class ProcessOwner

  Interface that allows the owner of a child process to be notified.
- class ChildProcessMgr

  Manage the child processes.
- class UDPSocket

  An abstract base class with common stuff for UDP sockets.
- class UDPClientSocket

  A UDP client socket.
- class UDPServerSocket

  A UDP Server socket.
- class XINI

  Load an XML configuration file.
- class Xml

  A C++ wrapper for libxml2.

Typedefs

- typedef std::shared_ptr
  < DiscoverPointer > DiscoverSharedPointer

  A shared pointer that will be owned by a discoverable object.
- typedef std::weak_ptr
  < DiscoverPointer > DiscoverWeakPointer

  A weak pointer that will be held by the DiscoveryMgr.
- typedef std::shared_ptr
  < PlugInLib > PlugInLibPtr

  Shared pointer to PlugInLib.
- typedef std::unique_ptr
  < AutoRunPlugIn > AutoRunPlugInPtr

  Unique pointer to an AutoRunPlugIn.

6.4.1 Detailed Description

The namespace for the Common Facilities Infrastructure components. The cfi namespace is for components that are basic infrastructure for almost any large project.

Chapter 7

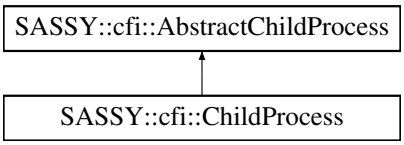
Class Documentation

7.1 SASSY::cfi::AbstractChildProcess Class Reference

Abstract child process interface for ChildProcessMgr.

`#include <cfi/proc.h>`

Inheritance diagram for SASSY::cfi::AbstractChildProcess:



Public Member Functions

- virtual ~AbstractChildProcess ()
  Destructor.
- virtual int getId ()=0
  Returns the pid of the child process.
- virtual void kill ()=0
  Terminates the child process.
- virtual bool done ()=0
  get the running state of the process.
- virtual void finished (int exit_status)=0
  Called by the manager when the child completes.

7.1.1 Detailed Description

Abstract child process interface for ChildProcessMgr.

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

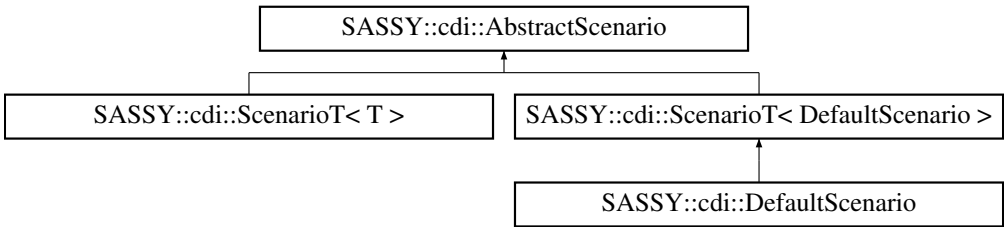
- proc.h

7.2 SASSY::cdi::AbstractScenario Class Reference

Define a type for scenarios.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::AbstractScenario:



Public Member Functions

- AbstractScenario (csr name)

  Constructor.
- virtual ~AbstractScenario ()

  Destructor.
- virtual bool willRunTests ()
- virtual void runTests (csr scenarioName, ScenarioResults *)

  Runs the tests within the scenario.

Protected Attributes

- std::string scenarioName

  The name of the scenario, as used when installed.

7.2.1 Detailed Description

Define a type for scenarios.

A scenario is responsible for setting up and taking down the resources required for a particular testing scenario.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 SASSY::cdi::AbstractScenario::AbstractScenario (csr *name*) [inline],[explicit]

Constructor.

Parameters

name	The name of the scenario.

7.2.3 Member Function Documentation

7.2.3.1 virtual bool SASSY::cdi::AbstractScenario::willRunTests ( ) [inline],[virtual]

Returns true if the scenario is going to manage the order in which the tests are run.

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

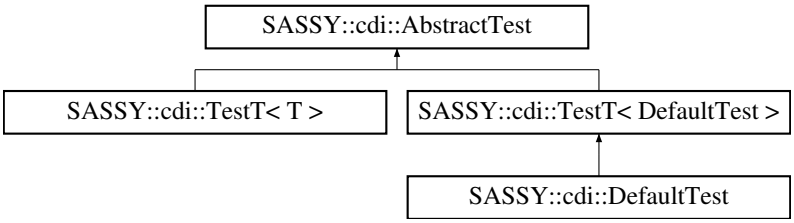
- testmgr.h

7.3 SASSY::cdi::AbstractTest Class Reference

Responsible for managing resources needed for the entire test.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::AbstractTest:



Public Member Functions

- virtual ~AbstractTest ()

  Destructor.

7.3.1 Detailed Description

Responsible for managing resources needed for the entire test.

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

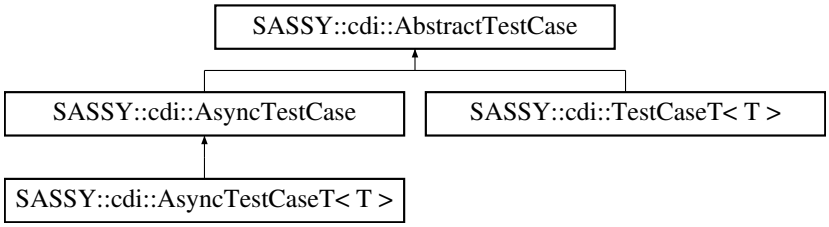
- testmgr.h

7.4 SASSY::cdi::AbstractTestCase Class Reference

Base class for test cases.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::AbstractTestCase:



Public Member Functions

- AbstractTestCase (csr nm)

  Constructor.
- virtual ~AbstractTestCase ()

  Destructor.
- virtual bool operator() (ScenarioResults *)

  Execute the tests.

Static Public Attributes

- static const bool EXPECTED = true

  Value for expected parameter of test()

Protected Member Functions

- bool test (bool cond, csr testMsg, bool expected=false)

  Perform or record a test.
- virtual bool runTest ()=0

  Run the tests for the test case.

Protected Attributes

- std::string name

  The name of the test case.
- ScenarioResults * scenario

  Results for the scenario are placed in here.

7.4.1 Detailed Description

Base class for test cases.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 SASSY::cdi::AbstractTestCase::AbstractTestCase (csr *nm*) [inline], [explicit]

Constructor.

Derived classes should perform any test case specific initialisations.

Parameters

nm	The name of the test case.

7.4.2.2 virtual SASSY::cdi::AbstractTestCase::~AbstractTestCase ( ) [inline], [virtual]

Destructor.

Derived classes should perform test case specific clean up.

7.4.3 Member Function Documentation

7.4.3.1 bool AbstractTestCase::operator() (ScenarioResults * *sr*) [virtual]

Execute the tests.

Returns

true if tests passed.

7.4.3.2 virtual bool SASSY::cdi::AbstractTestCase::runTest ( ) [protected], [pure virtual]

Run the tests for the test case.

Derived classes implement this to perform the required tests.

Returns

true if all tests passed.

7.4.3.3 bool AbstractTestCase::test (bool *cond*, csr *testMsg*, bool *expected* = false) [protected]

Perform or record a test.

Parameters

cond	Set to true if test passes.
testMsg	Message to report when test fails
expected	Set to EXPECTED if test is expected to fail.

Returns

true if test passes, or failed and was expected to fail

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

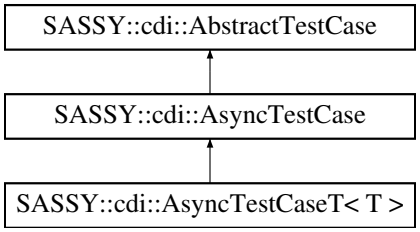
- testmgr.h
- testmgr.cpp

7.5 SASSY::cdi::AsyncTestCase Class Reference

Responsible for handling asynchronous tests.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::AsyncTestCase:



Public Member Functions

- AsyncTestCase (csr name)

  Constructor.

Protected Member Functions

- virtual void initTest ()=0

  Initiate the test case.

- virtual void handleEvent (TestEvent *ev)=0

  Handle the events.

- virtual void timedOut ()=0

  Handle time out.

Protected Attributes

- bool done

  This should be set by the handleEvent function.

- std::chrono::steady_clock::time_point startTime

  The time when the test is started.

- std::chrono::steady_clock::duration timeOut

  The time to wait. The default is 10 seconds.

Additional Inherited Members

7.5.1 Detailed Description

Responsible for handling asynchronous tests.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 AsyncTestCase::AsyncTestCase (csr *name*)

Constructor.

Parameters

name	The name of the test case.

7.5.3 Member Function Documentation

7.5.3.1 virtual void SASSY::cdi::AsyncTestCase::handleEvent (TestEvent * *ev*) [protected], [pure virtual]

Handle the events.

When the object under test gets its call back it will queue an event which is passed to this function. Set done to true when the last expected event arrives.

Parameters

ev	The test event.

7.5.3.2 virtual void SASSY::cdi::AsyncTestCase::initTest ( ) [protected], [pure virtual]

Initiate the test case.

The derived class should use this to start the test function. It should return immediately the test has been started.

7.5.3.3 virtual void SASSY::cdi::AsyncTestCase::timedOut ( ) [protected], [pure virtual]

Handle time out.

This gets called if the events don't arrive in time.

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

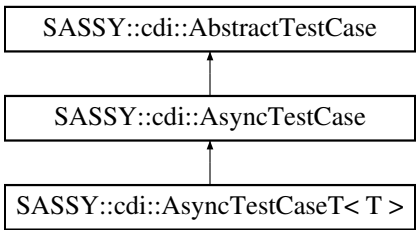
- testmgr.h
- testmgr.cpp

7.6 SASSY::cdi::AsyncTestCaseT< T > Class Template Reference

Responsible for installing the factory for the test case type.

`#include <testmgr.h>`

Inheritance diagram for SASSY::cdi::AsyncTestCaseT< T >:



Public Member Functions

- AsyncTestCaseT (csr nm)
  Constructor.

Static Public Member Functions

- static void install (csr nm, csr scn)
  Install the factory for the test case.

Additional Inherited Members

7.6.1 Detailed Description

template<class T>class SASSY::cdi::AsyncTestCaseT< T >

Responsible for installing the factory for the test case type.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 template<class T > SASSY::cdi::AsyncTestCaseT< T >::AsyncTestCaseT (csr *nm*) [inline]

Constructor.

Parameters

nm	The name of the test case

7.6.3 Member Function Documentation

7.6.3.1 template<class T > static void SASSY::cdi::AsyncTestCaseT< T >::install (csr *nm*, csr *scn*) [inline], [static]

Install the factory for the test case.

Parameters

nm	The name of the test case
scn	The name of the enclosing scenario.

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

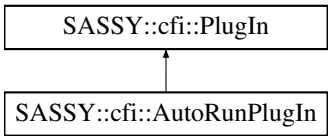
- testmgr.h

7.7 SASSY::cfi::AutoRunPlugIn Class Reference

Base class for plug ins that are run immediately that the library is loaded.

`#include <cfi/plugin.h>`

Inheritance diagram for SASSY::cfi::AutoRunPlugIn:



Public Member Functions

- virtual void execute ()=0

  Called by the PlugInMgr to run the plug-in's code.

7.7.1 Detailed Description

Base class for plug ins that are run immediately that the library is loaded.

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

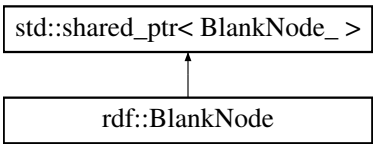
- plugin.h

7.8 rdf::BlankNode Class Reference

A shared pointer with constructors for the BlankNode_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::BlankNode:



Public Member Functions

- BlankNode (World, const std::string &id="")

  Create a blank node using a supplied identifier.
- BlankNode (Node blankNode)

  Create a BlankNode from a Node.

7.8.1 Detailed Description

A shared pointer with constructors for the BlankNode_ class.

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

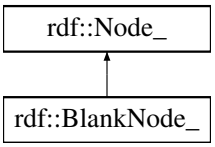
- rdfxx.h

7.9 rdf::BlankNode_ Class Reference

An abstract class defining the methods for an RDF blank Node.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::BlankNode_:



Public Member Functions

- virtual std::string toString () const =0

  Get a string representation of the node.
- virtual std::string toString (const Format &) const =0

  Get a string representation of the node using the supplied formatting instructions.

7.9.1 Detailed Description

An abstract class defining the methods for an RDF blank Node.

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

- rdfxx.h

7.10 SASSY::cdi::CallTrace Class Reference

Class to create a call trace.

`#include <cfi/trace.h>`

Public Member Functions

- CallTrace (int level, csr methodName, csr file)

  Constructor.
- ~CallTrace ()

  Destructor.

7.10.1 Detailed Description

Class to create a call trace.

This CallTrace class is used to produce a call graph. Use the macro FN_TRACE to create an object of this class on the stack at the beginning of each method. The destructor will be automatically called when the method ends.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 CallTrace::CallTrace (int *level*, csr *methodName*, csr *file*)

Constructor.

This should be used via the FN_TRACE(level,method) macro.

Parameters

level	used to determine if tracing output should be generated.
methodName	name of the function to report
file	name of the source file

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

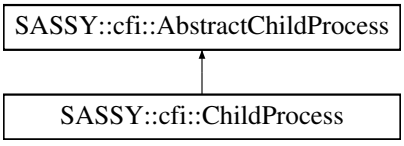
- trace.h
- trace.cpp

7.11 SASSY::cfi::ChildProcess Class Reference

Represents the state of a child process.

`#include <cfi/proc.h>`

Inheritance diagram for SASSY::cfi::ChildProcess:



Public Member Functions

- ChildProcess (const std::string &cmnd, int *fdin=nullptr, int *fdout=nullptr)
  constructor
- ChildProcess ()
  constructor for fork
- virtual ~ChildProcess ()
  destructor
- virtual void setArgs (const std::vector< std::string > &args)
  set the args for the process
- virtual void run ()
  start the command running
- virtual void kill ()
  terminate the process
- virtual int signal (int sig)
  send a signal to the process
- virtual void finished (int result)
  set the exit status of the process
- bool done ()
  get the running status of the program
- virtual int getId ()
  get the process id
- int getExitSignal ()
  get the signal that caused the process to exit.

7.11.1 Detailed Description

Represents the state of a child process.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 ChildProcess::ChildProcess (const std::string & *cmnd*, int * *fdin* = `nullptr`, int * *fdout* = `nullptr`)

constructor

Parameters

cmnd	the command to run
fdin	File descriptor for stdin for the child
fdout	File descriptor for stdout for the child

7.11.3 Member Function Documentation

7.11.3.1 void ChildProcess::finished (int *result*) `[virtual]`

set the exit status of the process

called by the process manager when the process exits

Parameters

result	the exit status and code of the process

Implements SASSY::cfi::AbstractChildProcess.

7.11.3.2 void ChildProcess::setArgs (const std::vector< std::string > & *args*) [virtual]

set the args for the process

Parameters

args	the args for the command

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

- proc.h
- proc.cpp

7.12 SASSY::cfi::ChildProcessMgr Class Reference

Manage the child processes.

`#include <cfi/proc.h>`

Public Member Functions

- void registerOwner (ProcessOwner *)

  register an object that is to be notified of a child's death
- void registerChild (AbstractChildProcess *cp)

  tell the manager about a child
- void deregisterChild (AbstractChildProcess *cp)

  forget about a child
- int numberOfChildren () const

  get the number of child processes
- int numberOfLiveChildren () const

  get the number of running children
- void shutDown ()

  terminate all the child processes

Static Public Member Functions

- static ChildProcessMgr & instance ()

  get reference to the child process manager

7.12.1 Detailed Description

Manage the child processes.

Manage the child processes at a global level which is mostly about handling the SIGCHLD signals and ensuring the correct ChildProcess gets notified.

7.12.2 Member Function Documentation

7.12.2.1 void ChildProcessMgr::deregisterChild (AbstractChildProcess * *cp*)

forget about a child

called by the ChildProcess as it destructs

Parameters

cp	the child process

7.12.2.2 int SASSY::cfi::ChildProcessMgr::numberOfChildren ( ) const [inline]

get the number of child processes

Returns

the number of child processes

7.12.2.3 int ChildProcessMgr::numberOfLiveChildren ( ) const

get the number of running children

Returns

the number of children still running

7.12.2.4 void ChildProcessMgr::registerChild (AbstractChildProcess * *cp*)

tell the manager about a child

called by the ChildProcess as it creates the process

Parameters

cp	the child process

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

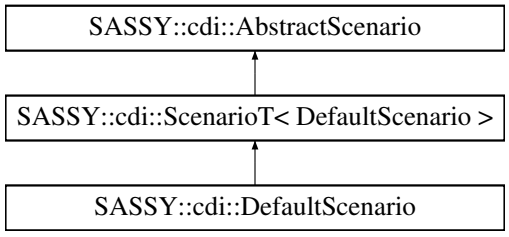
- proc.h
- proc.cpp

7.13 SASSY::cdi::DefaultScenario Class Reference

Provide a default scenario object.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::DefaultScenario:



Public Member Functions

- DefaultScenario (csr name)

  Constructor.
- DefaultScenario ()

  Constructor.
- ~DefaultScenario ()

  Destructor.

Additional Inherited Members

7.13.1 Detailed Description

Provide a default scenario object.

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

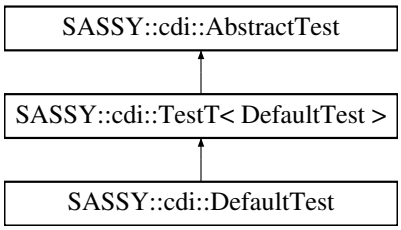
- testmgr.h

7.14 SASSY::cdi::DefaultTest Class Reference

Provide a default version of the AbstractTest object.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::DefaultTest:



Public Member Functions

- DefaultTest ()

  Default constructor that does nothing.
- ~DefaultTest ()

  Destructor.

Additional Inherited Members

7.14.1 Detailed Description

Provide a default version of the AbstractTest object.

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

- testmgr.h

7.15 SASSY::cfi::Discoverable Class Reference

A mixin class that makes its owner discoverable.

`#include <cfi/discover.h>`

Protected Attributes

- DiscoverSharedPointer discover
  A shared pointer that references back to the discoverable object.

7.15.1 Detailed Description

A mixin class that makes its owner discoverable.

Adds a shared pointer to a DiscoverPointer to an object so that it can be discovered.

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

- discover.h

7.16 SASSY::cfi::DiscoverPointer Struct Reference

Holds a pointer to a discoverable object.

`#include <cfi/discover.h>`

Public Member Functions

- DiscoverPointer ()
  Default constructor.
- DiscoverPointer (void *a)
  Constructor.

Public Attributes

- void * addr
  Pointer to a discoverable object.

7.16.1 Detailed Description

Holds a pointer to a discoverable object.

A discoverable object holds a shared pointer to this object. The DiscoveryManager holds a weak pointer to this object so that it can tell when the owner has been deleted.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 SASSY::cfi::DiscoverPointer::DiscoverPointer (void * *a*) [inline]

Constructor.

Parameters

a	The address of the discoverable object.

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

- discover.h

7.17 SASSY::cfi::DiscoveryMgr Class Reference

Manager for discoverable objects.

`#include <cfi/discover.h>`

Public Member Functions

- void save (const char *prettyName, DiscoverSharedPointer p)
  Save a discoverable object.
- void fetch (csr name, std::vector< void * > &results)
  Get a list of objects with a class name.

Static Public Member Functions

- static DiscoveryMgr & instance ()
  Return a reference to the DiscoveryMgr.

7.17.1 Detailed Description

Manager for discoverable objects.

The manager keeps a list of discoverable objects and is responsible for supplying a list of objects given a class name.

7.17.2 Member Function Documentation

7.17.2.1 void DiscoveryMgr::fetch (csr *name*, std::vector< void * > & *results*)

Get a list of objects with a class name.

Parameters

name	The class name to search for.
results	A vector which is filled in with pointers to the objects.

7.17.2.2 void DiscoveryMgr::save (const char * *prettyName*, DiscoverSharedPointer *p*)

Save a discoverable object.

Parameters

prettyName	The name of the constructor as provided by **PRETTY_FUNCTION**
p	A shared pointer to the discoverable object.

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

- discover.h
- discover.cpp

7.18 rdf::ErrorClient Class Reference

Client that is notified of errors and/or warnings.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~ErrorClient ()
  Virtual destructor.
- virtual void handleError (const std::string &message)=0
  The function that will be called when an error is detected.
- virtual void handleWarning (const std::string &message)=0
  The function that will be called when a warning is detected.

7.18.1 Detailed Description

Client that is notified of errors and/or warnings.

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

- rdfxx.h

7.19 SASSY::cfi::FD Class Reference

Wrap file descriptors in a class to ensure closed.

`#include <sos/fdstream.h>`

Public Member Functions

- FD (const std::string &name, int mode, bool lock)
  constructor which opens the file
- FD (int x)
  constructor for existing file descriptor
- ~FD ()
  destructor
- operator int ()
  type conversion

Static Public Member Functions

- static void report (std::ostream &s, int fd)
  Report the details or status of a file descriptor.

7.19.1 Detailed Description

Wrap file descriptors in a class to ensure closed.

The FD class is used to wrap a file descriptor so that it will be automatically closed when it goes out of scope. This ensures that file descriptors are closed when an exception is thrown.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 FD::FD (const std::string & *name*, int *mode*, bool *lock*)

constructor which opens the file

Parameters

name	the name of file to open
mode	the open mode
lock	if true the file is also locked

7.19.2.2 SASSY::cfi::FD::FD (int *x*) [inline]

constructor for existing file descriptor

Parameters

x	the existing file descriptor

7.19.3 Member Function Documentation

7.19.3.1 void FD::report (std::ostream & *s*, int *fd*) [static]

Report the details or status of a file descriptor.

Parameters

s	the stream to write to
fd	the file descriptor to examine.

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

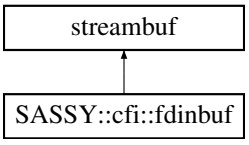
- fdstream.h
- fd.cpp

7.20 SASSY::cfi::fdinbuf Class Reference

An input stream buffer.

`#include <sos/fdstream.h>`

Inheritance diagram for SASSY::cfi::fdinbuf:



Public Member Functions

- fdinbuf (int fd, bool isCloseNeeded, fdistream &owner)

  constructor
- void close ()

  Close the input file descriptor.
- virtual ~fdinbuf ()

  destructor.

Protected Member Functions

- virtual int underflow ()

  read characters from the buffer.

Protected Attributes

- fdistream & mOwner

  Our owning input stream.
- int mFd

  The input file descriptor.
- bool mIsCloseNeeded

  Whether the file descriptor needs to be closed on destruction.
- char mBuffer [BUFFER_SIZE+PUSHBACK_SIZE]

Static Protected Attributes

- static const int BUFFER_SIZE = 8192

  Size of buffer.
- static const int PUSHBACK_SIZE = 4

  maximum number of push back characters supported

7.20.1 Detailed Description

An input stream buffer.

An input stream buffer which reads from a file descriptor. This provides input buffering.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 fdinbuf::fdinbuf (int *fd*, bool *isCloseNeeded*, fdistream & *owner*)

constructor

Constructor which accepts the file descriptor to read from.

Parameters

fd	the file descriptor to read from
isCloseNeeded	a flag indicating whether the file descriptor is closed on destruction

owner	our owning input stream

7.20.3 Member Data Documentation

7.20.3.1 char SASSY::cfi::fdinbuf::mBuffer[BUFFER_SIZE+PUSHBACK_SIZE] [protected]

Input buffer. This holds the data read and a bit more for a push-back buffer.

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

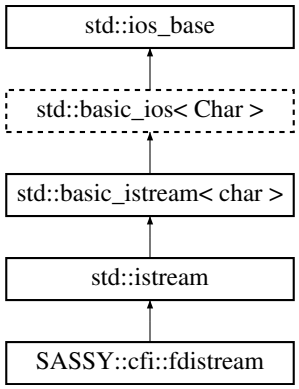
- fdstream.h
- fdistream.cpp

7.21 SASSY::cfi::fdistream Class Reference

A file descriptor input stream.

`#include <sos/fdstream.h>`

Inheritance diagram for SASSY::cfi::fdistream:



Public Member Functions

- fdistream (int fd, bool isCloseNeeded=false)
  constructor
- void close ()
  close the file descriptor

Static Public Attributes

- static bool CLOSE_NEEDED = true
  Flag to indicate file descriptor should be closed.
- static bool CLOSE_NOT_NEEDED = false
  Flag to indicate file descriptor should not be closed.

Protected Attributes

- fdinbuf mBuf
  Our input file descriptor stream buffer.

Friends

- class **fdinbuf**

7.21.1 Detailed Description

A file descriptor input stream.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 fdistream::fdistream (int *fd*, bool *isCloseNeeded* = `false`)

constructor

Constructor which accepts the file descriptor to read from

Parameters

fd	the file descriptor to read from
isCloseNeeded	a flag indicating whether the file descriptor is closed on destruction.

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

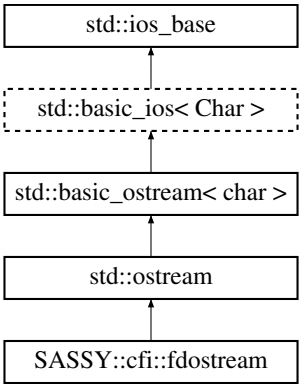
- fdstream.h
- fdistream.cpp

7.22 SASSY::cfi::fdostream Class Reference

A file descriptor output stream.

`#include <sos/fdstream.h>`

Inheritance diagram for SASSY::cfi::fdostream:



Public Member Functions

- fdostream (int fd, bool isCloseNeeded=false)

  constructor
- void close ()

  close the file descriptor

Static Public Attributes

- static bool CLOSE_NEEDED = true

  Flag to indicate file descriptor should be closed.
- static bool CLOSE_NOT_NEEDED = false

  Flag to indicate file descriptor should not be closed.

Protected Attributes

- fdoutbuf mBuf

  Our output file descriptor stream buffer.

Friends

- class **fdoutbuf**

7.22.1 Detailed Description

A file descriptor output stream.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 fdostream::fdostream (int *fd*, bool *isCloseNeeded* = `false`)

constructor

Constructor which accepts the file descriptor to write to and a flag indicating whether the file descriptor is closed on destruction.

Parameters

fd	the file descriptor to write to
isCloseNeeded	if true close on destruction

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

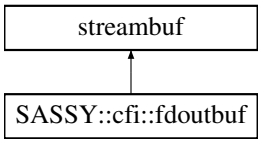
- fdstream.h
- fdostream.cpp

7.23 SASSY::cfi::fdoutbuf Class Reference

An output stream buffer.

`#include <sos/fdstream.h>`

Inheritance diagram for SASSY::cfi::fdoutbuf:



Public Member Functions

- fdoutbuf (int fd, bool isCloseNeeded, fdostream &owner)

  constructor
- void close ()

  Close the output file descriptor.
- virtual ~fdoutbuf ()

  destructor.

Protected Member Functions

- int flushBuffer ()

  Flush the characters in the buffer.
- virtual int overflow (int c)

  Write the indicated character and all previous characters.
- virtual int sync ()

  Flush the data in the buffer.

Protected Attributes

- fdostream & mOwner

  Our owning output stream.
- int mFd

  The output file descriptor.
- bool mIsCloseNeeded

  Whether the file descriptor needs to be closed on destruction.
- char mBuffer [BUFFER_SIZE]

  Output buffer.

Static Protected Attributes

- static const int BUFFER_SIZE = 8192

  Output buffer size.

7.23.1 Detailed Description

An output stream buffer.

An output stream buffer which writes to a file descriptor. This provides output buffering.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 fdoutbuf::fdoutbuf (int *fd*, bool *isCloseNeeded*, fdostream & *owner*)

constructor

Parameters

fd	the file descriptor to write to
isCloseNeeded	a flag indicating whether the file descriptor is closed on destruction
owner	our owning output stream

7.23.3 Member Function Documentation

7.23.3.1 void fdoutbuf::close ( )

Close the output file descriptor.

This will be done whether or not the close needed flag is set.

7.23.3.2 int fdoutbuf::overflow (int *c*) [protected], [virtual]

Write the indicated character and all previous characters.

Parameters

c	the current character.

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

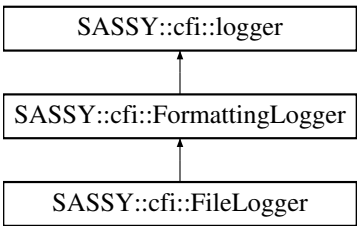
- fdstream.h
- fdostream.cpp

7.24 SASSY::cfi::FileLogger Class Reference

Class for logging to a file.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::FileLogger:



Public Member Functions

- FileLogger (const Path &fname)
  Constructor.
- int log (Severity, csr line)
  log to the unnamed log
- int log (csr logName, Severity, csr line)
  log to the named log

Protected Attributes

- std::ofstream f
  The logging stream.

Additional Inherited Members

7.24.1 Detailed Description

Class for logging to a file.

Writes log messages to a file.

The log messages include the output from the FormattingLogger.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 FileLogger::FileLogger (const Path & *fname*)

Constructor.

Parameters

fname	the name of the log file.

7.24.3 Member Function Documentation

7.24.3.1 int FileLogger::log (Severity *sev*, csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.24.3.2 int FileLogger::log (csr *logName*, Severity *sev*, csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

- log.h
- log.cpp

7.25 rdf::Format Struct Reference

Instructions for converting a node to a string.

`#include <rdfxx/rdfxx.h>`

Public Attributes

- bool usePrefixes
  Replace namespaces with prefixes.
- bool angleBrackets
  Enclose URI in '<' ... '>'.
- std::string blank
  Name to give blank nodes.
- bool quotes
  Put values in quotes.
- bool showLanguage
  Include language identifier as for example: "@en".
- std::string prefLang
  Restrict to this language when there are alternatives.
- bool showDataType
  Include data type as for example: ^^xsd:integer.

7.25.1 Detailed Description

Instructions for converting a node to a string.

These objects are passed to the toString() method of Nodes and Statements to control how they will be presented.

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

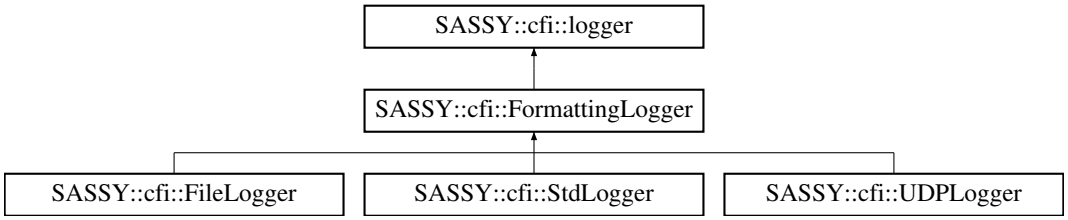
- rdfxx.h

7.26 SASSY::cfi::FormattingLogger Class Reference

Class for producing a formatted log message.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::FormattingLogger:



Protected Member Functions

- std::string timeStamp ()
  generates a string containing the current time

- csr hostname ()

  generates a string containing the hostname.
- csr process ()

  generates a string containing the process id
- csr severity (Severity)

  generates a string containing the severity level
- int format (std::ostream &s, Severity sev, csr msg)

  output the message to the stream

Additional Inherited Members

7.26.1 Detailed Description

Class for producing a formatted log message.

Provides a common formatting function for the different loggers. Writes log messages with a time stamp, host name, and process id in addition to the severity and the message.

7.26.2 Member Function Documentation

7.26.2.1 int FormattingLogger::format (std::ostream & *s*, Severity *sev*, csr *msg*) [protected]

output the message to the stream

Parameters

s	the log stream to write to
sev	the severity level to use
msg	the message to write to the log

Returns

the number of characters written to the stream

7.26.2.2 string FormattingLogger::timeStamp ( ) [protected]

generates a string containing the current time

Returns

current time as a string with format of H:M:S Y-m-d

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

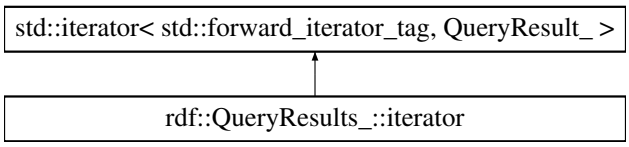
- log.h
- log.cpp

7.27 rdf::QueryResults_::iterator Class Reference

Provide a C++ iterator over the results of a RDF query.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::QueryResults_::iterator:



Public Member Functions

- iterator (QueryResult_ *qr)
  Construct an iterator over the results.
- iterator ()
  Default constructor.
- iterator & operator= (const QueryResult_ &)=delete
  No assignment allowed for these iterators.
- virtual QueryResult_ & operator* () const
  Dereference the iterator to get a result.
- virtual iterator & operator++ ()
  Move to next result.
- virtual iterator & operator++ (int)
  Move to next result.
- virtual bool operator== (const iterator &) const
  Check for equality.
- virtual bool operator!= (const iterator &) const
  Check for inequality.

7.27.1 Detailed Description

Provide a C++ iterator over the results of a RDF query.

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

- rdfxx.h

7.28 rdf::Literal Class Reference

Hold the value, language and data type for an RDF literal.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- Literal ()
  Default constructor.
- Literal (const std::string &val)
  Create a PlainLiteral with English language.
- Literal (const char *val)
  Create a PlainLiteral with English language.
- Literal (const std::string &val, const std::string &lan)
  Create a PlainLiteral with the specified language.
- Literal (const std::string &val, DataType, const std::string &lan="en")

Create a literal with the specified content, data type and/or language.

- Literal (int, DataType)

  Create a literal with some type of integral value.
- Literal (double, DataType)

  Create a literal with some type of real number value.
- Literal (bool)

  Create a literal with a boolean value.
- void language (const std::string &lang)

  Set the language.
- void dataType (DataType t)

  Set the data type.
- void value (const std::string &v)

  Set the value.
- std::string toString () const

  Default conversion to a string.
- std::string toString (const Format &) const

  Convert to astringusing the specified format.
- DataType dataType () const

  Get the data type.
- URI dataTypeURI (World) const

  Get the data type as a URI.
- std::string language () const

  Get the language.
- std::string asString () const

  Get the value as a string.
- int asInteger () const

  Get the value as an integer.
- double asDouble () const

  Get the value as a double.
- bool asBoolean () const

  Get the value as a boolean.

Static Public Member Functions

- static std::string toXSD (DataType)

  Convert a data type into an xsd form,.
- static std::string **toTypeName** (DataType)
- static DataType toDataType (const std::string &xsd_type)

  Convert an xsd form into a data type.
- static DataType **asDataType** (const std::string &type_name)
- static std::vector< std::string > **getDataTypeNames** ()

7.28.1 Detailed Description

Hold the value, language and data type for an RDF literal.

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

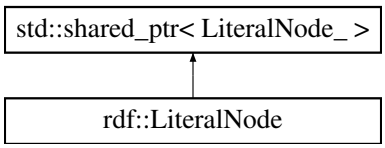
- rdfxx.h

7.29 rdf::LiteralNode Class Reference

A shared pointer with constructors for the LiteralNode_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::LiteralNode:



Public Member Functions

- LiteralNode (World, const Literal &)

  Create a literal node using a literal value.
- LiteralNode (Node literalNode)

  Convert a Node to a LiteralNode.

7.29.1 Detailed Description

A shared pointer with constructors for the LiteralNode_ class.

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

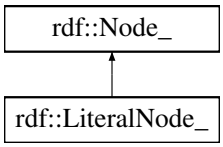
- rdfxx.h

7.30 rdf::LiteralNode_ Class Reference

An abstract class defining the methods for an RDF Literal Node.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::LiteralNode_:



Public Member Functions

- virtual std::string toString () const =0

  Get a string representation of the node.
- virtual std::string toString (const Format &) const =0

  Get a string representation of the node using the supplied formatting instructions.
- virtual Literal toLiteral () const =0

  Get the Literal data for the node.

7.30.1 Detailed Description

An abstract class defining the methods for an RDF Literal Node.

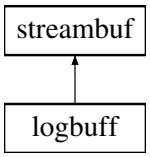
The documentation for this class was generated from the following file:

- rdfxx.h

7.31 logbuff Class Reference

A streambuf class specialized for logging.

Inheritance diagram for logbuff:



Public Member Functions

- logbuff ()
  default constructor
- logbuff (csr)
  constructor for named log
- virtual ∼logbuff ()
  destructor
- void severity (Severity s)
  set severity

Protected Member Functions

- void init ()
  initialize using details from the XINI config
- int flushBuffer ()
- virtual int overflow (int c)
- virtual int sync ()

Protected Attributes

- char buffer [bufferSize]
  message collected into here
- Severity sev
  the current severity level
- logger ∗ mLogger
  thing for writing the message
- string name
  name to use in the XINI configuration

Static Protected Attributes

- static const int bufferSize = 128

  limit line size in log files.

7.31.1 Detailed Description

A streambuf class specialized for logging.

This is a private implementation class.

7.31.2 Member Function Documentation

7.31.2.1 int logbuff::flushBuffer ( ) [protected]

we have a line for the log - we now delegate this to the log function that has been selected by the user.

7.31.2.2 int logbuff::overflow (int *c*) [protected], [virtual]

this gets called when the buffer fills. Since we are writing to a log file, this happens whenever the line length exceeds the buffer size.

7.31.2.3 int logbuff::sync ( ) [protected], [virtual]

this gets called when 'endl' or 'flush' is called on the stream.

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

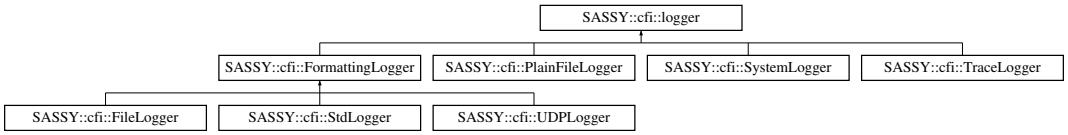
- log.cpp

7.32 SASSY::cfi::logger Class Reference

An abstract base class used by the log stream to write logs.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::logger:



Public Member Functions

- virtual ~logger ()

  Destructor.
- virtual int log (Severity sev, csr line)=0

  log to the unnamed log
- virtual int log (csr logName, Severity sev, csr line)=0

  log to the named log

Protected Member Functions

- logger ()

  Constructor.

Protected Attributes

- Semaphore sem4

  Mutual exclusion lock.

7.32.1 Detailed Description

An abstract base class used by the log stream to write logs.

This is the API that libraries must implement to be used as a logging stream.

7.32.2 Member Function Documentation

7.32.2.1 virtual int SASSY::cfi::logger::log (Severity *sev,* csr *line*) [pure virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implemented in SASSY::cfi::SystemLogger, SASSY::cfi::TraceLogger, SASSY::cfi::UDPLogger, SASSY::cfi::PlainFileLogger, SASSY::cfi::FileLogger, and SASSY::cfi::StdLogger.

7.32.2.2 virtual int SASSY::cfi::logger::log (csr *logName,* Severity *sev,* csr *line*) [pure virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implemented in SASSY::cfi::SystemLogger, SASSY::cfi::TraceLogger, SASSY::cfi::UDPLogger, SASSY::cfi::PlainFileLogger, SASSY::cfi::FileLogger, and SASSY::cfi::StdLogger.

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

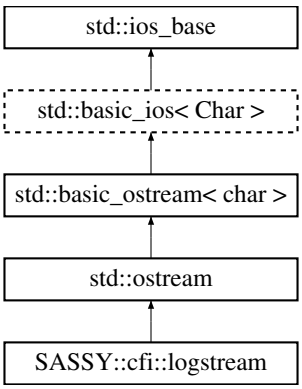
- log.h

7.33 SASSY::cfi::logstream Class Reference

A stream class specialized for logging.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::logstream:



Public Member Functions

- virtual ~logstream ()

  destructor
- void severity (Severity sev)

  Set the severity for the current message.

Static Public Member Functions

- static logstream & instance ()

  Get an instance of the unnamed stream.
- static logstream & instance (csr name)

  Get an instance of a named stream.

Protected Member Functions

- logstream (std::streambuf *buf)

  Static pointer to only instance of the anonymous object.

7.33.1 Detailed Description

A stream class specialized for logging.

There is an unnamed stream that is the default, and named streams for special purposes. The streams pick up their characteristics from XINI configuration file:

- //LOG the logging section of the configuration file
- //LOG/[name] section for a named log
- //LOG/type is one of cout, cerr, clog, file, trace or syslog.
- //LOG/file specifies the file name for file or trace logs
- //LOG/ident specifies the identifier for syslog logs

7.33.2 Constructor & Destructor Documentation

7.33.2.1 logstream::logstream (std::streambuf * *buf*) [protected]

Static pointer to only instance of the anonymous object.

Single instances of named streams. Private constructor.

Parameters

buf	pointer to a standard stream buffer

7.33.3 Member Function Documentation

7.33.3.1 logstream & logstream::instance ( ) [static]

Get an instance of the unnamed stream.

Returns

the instance of the anonymous log stream

7.33.3.2 logstream & logstream::instance (csr *name*) [static]

Get an instance of a named stream.

Parameters

name	name of the log stream used in the config file

Returns

the instance of the named log stream

7.33.3.3 void logstream::severity (Severity *sev*)

Set the severity for the current message.

Parameters

sev	new severity to use for the following messages

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

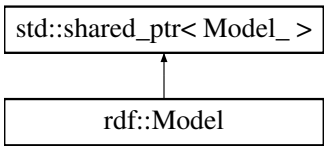
- log.h
- log.cpp

7.34 rdf::Model Class Reference

A shared pointer with constructors for the Model_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Model:



Public Member Functions

- Model ()

  Create an empty model.
- Model (Model_ *)

  Replicate shared_ptr<> constructor.
- Model (World, const std::string &storage_type, const std::string &storage_name="", const std::string &storage_options="", const std::string &model_options="")

  New model attached to storage.

7.34.1 Detailed Description

A shared pointer with constructors for the Model_ class.

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

- rdfxx.h

7.35 rdf::Model_ Class Reference

An abstract class defining the methods for an RDF Model.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Model_ ()

  Virtual destructor.
- virtual World getWorld ()=0

  Return a reference to the model's world.
- virtual int size () const =0

  Get number of statements if possible. May return <0 if not known.
- virtual bool sync ()=0

  Save the model to its storage.
- virtual Stream toStream ()=0

  Get a pointer to a stream. The user controls its lifetime.
- virtual bool add (Node subject, Node predicate, Node object)=0

  Add the nodes of a statement to the model.
- virtual bool add (Statement)=0

  Add a statement to the model.
- virtual bool remove (Statement)=0

  Remove a statement from the model.
- virtual bool update (Statement old, Statement _new)=0

  Update a statement in the model.
- virtual bool contains (Statement) const =0

Check if a statement is in the model.

- virtual std::vector< Node > predicates (Node subject, Node object)=0

  Get a list of predicates that link a subject and object.

- virtual std::vector< Node > objects (Node subject, Node predicate)=0

  Get a list of objects that are linked to a subject by a particular predicate.

- virtual std::vector< Node > subjects (Node predicate, Node object)=0

  Get a list of subjects that are linked to an object by a particular predicate.

- virtual std::vector< Node > arcsIn (Node object)=0

  Get a list of predicates connected to an object.

- virtual std::vector< Node > arcsOut (Node subject)=0

  Get a list of predicates connected to a ssubject.

7.35.1 Detailed Description

An abstract class defining the methods for an RDF Model.

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

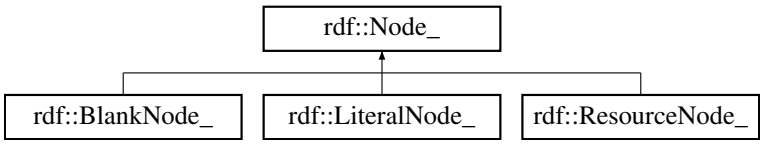
- rdfxx.h

7.36 rdf::Node_ Class Reference

An abstract class defining the methods for an RDF Node.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Node_:



Public Member Functions

- virtual ~Node_ ()

  Virtual destructor.

- virtual std::string toString () const =0

  Get a string representation of the node.

- virtual std::string toString (const Format &) const =0

  Get a string representation of the node using the supplied formatting instructions.

- virtual URI toURI () const =0

  Get the URI for a Node.

- virtual bool isLiteral () const =0

  Check if the node is a literal.

- virtual bool isBlank () const =0

  Check if the node is blank.

- virtual bool isResource () const =0

  Check if the node is a resource.

7.36.1 Detailed Description

An abstract class defining the methods for an RDF Node.

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

- rdfxx.h

7.37 SASSY::cdi::NullTrace Class Reference

A class for dummy trace objects.

`#include <cfi/trace.h>`

Public Member Functions

- NullTrace ()

  Constructor.
- template<class T >
  NullTrace & operator<< (T x)

  Noop version of stream function.

7.37.1 Detailed Description

A class for dummy trace objects.

A dummy object is required so that stream syntax can be used without causing a problem if tracing is turned off.

This is used by the TRACE(level) macro.

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

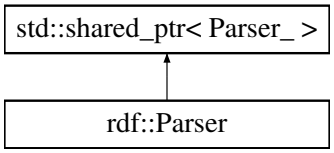
- trace.h

7.38 rdf::Parser Class Reference

A shared pointer with constructors for the Parser_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Parser:



Public Member Functions

- Parser (World, const std::string &name, const std::string &syntax_mime=std::string())

  Create an RDF parser using the specified parsing engine.
- Parser (World, const std::string &name, URI syntax_uri)

  Create an RDF parser using the specified parsing engine.

7.38.1 Detailed Description

A shared pointer with constructors for the Parser_ class.

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

- rdfxx.h

7.39 rdf::Parser_ Class Reference

An abstract class defining the methods for an RDF Parser.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Parser_ ()

  Virtual destructor.
- virtual bool parseIntoModel (Model, URI uri, URI base_uri)=0

  Parse a data source into a model.

Static Public Member Functions

- static std::vector< std::string > listParsers (World)

  Get a list of parser names with their syntax URIs.

7.39.1 Detailed Description

An abstract class defining the methods for an RDF Parser.

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

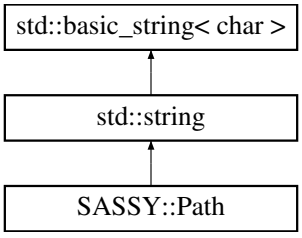
- rdfxx.h

7.40 SASSY::Path Class Reference

Manipulate path strings.

`#include <cfi/stringy.h>`

Inheritance diagram for SASSY::Path:



Public Member Functions

- Path ()
  Constructor.
- Path (csr path)
  Constructor.
- Path & append (csr name)
  Append aname to a path.
- Path & appendExt (csr ext)
  Append an extension to this.
- Path & up ()
  shorten this by a level
- bool absolute () const
- std::string base () const
- std::string dir () const
- std::string ext () const
- std::string name () const
- std::string noExt () const
- void split (std::vector< std::string > &dirs) const
  break into separate dirs
- bool operator== (const Path &x) const
  test for equality
- bool isChildOf (const Path &p) const
  test for equality
- Path & operator= (const char *)
  assignment
- Path & operator= (const std::string &)
  assignment
- operator const char * () const
  type conversion
- bool defined () const
  type conversion

Static Public Member Functions

- static bool check (const std::string &path)
  checks for valid string
- static void setSpacePolicy (bool allowSpaces=false)
  set policy for spaces in file names

7.40.1 Detailed Description

Manipulate path strings.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 Path::Path (csr *path*) [explicit]

Constructor.

normalizes out any surplus /'s

Parameters

path	path to a file or directory

7.40.3 Member Function Documentation

7.40.3.1 bool Path::absolute ( ) const

Returns

true if its an absolute path

7.40.3.2 Path & Path::append (csr *name*)

Append aname to a path.

Parameters

name	append a / and name to this

Returns

reference to new value of this

7.40.3.3 Path & Path::appendExt (csr *ext*)

Append an extension to this.

Parameters

ext	the extension to append.

Returns

reference to new value of this

7.40.3.4 string Path::base ( ) const

Returns

everything after last /

7.40.3.5 bool Path::defined ( ) const

type conversion

true if it has a value

7.40.3.6 string Path::dir ( ) const

Returns

everything up to last /

7.40.3.7 string Path::ext ( ) const

Returns

extension part of file name

7.40.3.8 bool Path::isChildOf (const Path & *p*) const

test for equality

test for equality true if this is a subdir of p

7.40.3.9 string Path::name ( ) const

Returns

base without the extension

7.40.3.10 string Path::noExt ( ) const

Returns

everything but extension

7.40.3.11 void Path::split (std::vector< std::string > & *dirs*) const

break into separate dirs

Parameters

dirs	The directory parts of the file name returned.

7.40.3.12 Path & Path::up ( )

shorten this by a level

Returns

reference to new value of this

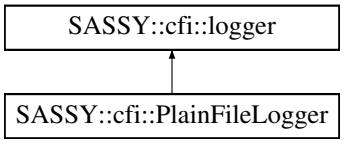
The documentation for this class was generated from the following files:

- stringy.h
- stringy.cpp

7.41 SASSY::cfi::PlainFileLogger Class Reference

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::PlainFileLogger:



Public Member Functions

- PlainFileLogger (const Path &fname)

  Constructor.
- int log (Severity, csr line)

  log to the unnamed log
- int log (csr logName, Severity, csr line)

  log to the named log

Protected Attributes

- std::ofstream f

  The logging stream.

Additional Inherited Members

7.41.1 Detailed Description

Writes log messages to a file without adding any formatting.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 PlainFileLogger::PlainFileLogger (const Path & *fname*)

Constructor.

Parameters

fname	the name of the log file.

7.41.3 Member Function Documentation

7.41.3.1 int PlainFileLogger::log (Severity *sev*, csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.41.3.2 int PlainFileLogger::log (csr *logName,* Severity *sev,* csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

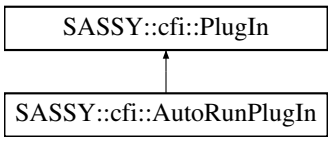
- log.h
- log.cpp

7.42 SASSY::cfi::PlugIn Class Reference

Base class for objects loaded from dynamically loaded libraries.

`#include <cfi/plugin.h>`

Inheritance diagram for SASSY::cfi::PlugIn:



Public Member Functions

- virtual ~PlugIn ()

  Destructor.

Friends

- class **PlugInMgr**

7.42.1 Detailed Description

Base class for objects loaded from dynamically loaded libraries.

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

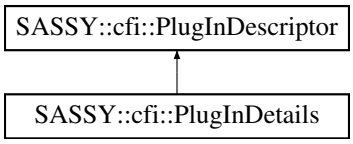
- plugin.h
- plugin.cpp

7.43 SASSY::cfi::PlugInDescriptor Struct Reference

Descriptor for plug-ins that can be used by the application.

`#include <plugin.h>`

Inheritance diagram for SASSY::cfi::PlugInDescriptor:



Public Attributes

- std::string pluginFamily

  The type of the plug-in.
- std::string name

  The name as in the xini config.
- std::string description

  Some description for users.
- std::string version

  Should match the defined SASSY_PLUGIN_VERSION.
- SASSY::Path library

  The shared library's path.
- bool onDemand

  True if the library is loaded on demand.
- bool autoRun

  True if its run automatically on start.

7.43.1 Detailed Description

Descriptor for plug-ins that can be used by the application.

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

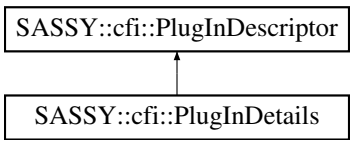
- plugin.h

7.44 SASSY::cfi::PlugInDetails Struct Reference

Details of plug-ins as provided by the loaded library.

`#include <plugin.h>`

Inheritance diagram for SASSY::cfi::PlugInDetails:



Public Member Functions

- PlugInDetails ()

  Constructor to initialise variables.

Public Attributes

- PlugInFactory * factory

  Construct an instance of the plug-in.
- bool loadable

  False if the library cannot be loaded.

7.44.1 Detailed Description

Details of plug-ins as provided by the loaded library.

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

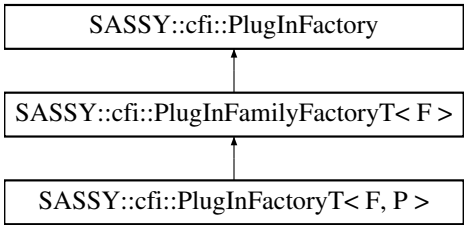
- plugin.h
- plugin.cpp

7.45 SASSY::cfi::PlugInFactory Class Reference

Base class for factories that create plug-in objects.

`#include <cfi/plugin.h>`

Inheritance diagram for SASSY::cfi::PlugInFactory:



Public Member Functions

- virtual ~PlugInFactory ()

  Destructor.
- virtual PlugIn * make ()=0

  Construct a plug-in object.

7.45.1 Detailed Description

Base class for factories that create plug-in objects.

7.45.2 Member Function Documentation

7.45.2.1 virtual PlugIn* SASSY::cfi::PlugInFactory::make ( ) [pure virtual]

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implemented in SASSY::cfi::PlugInFactoryT< F, P >, and SASSY::cfi::PlugInFamilyFactoryT< F >.

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

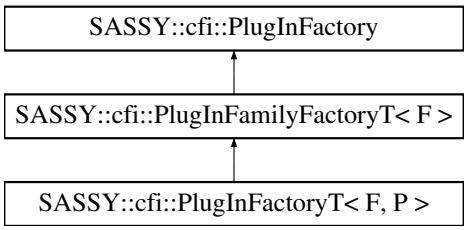
- plugin.h

7.46 SASSY::cfi::PlugInFactoryT< F, P > Class Template Reference

Template class for factories.

`#include <cfi/plugin.h>`

Inheritance diagram for SASSY::cfi::PlugInFactoryT< F, P >:



Public Member Functions

- virtual ~PlugInFactoryT ()
  Destructor.
- virtual F * make ()
  Construct a plug-in object.

7.46.1 Detailed Description

template<class F, class P>class SASSY::cfi::PlugInFactoryT< F, P >

Template class for factories.

PlugIn P must be derived from PlugInFamily F

7.46.2 Member Function Documentation

7.46.2.1 template<class F , class P > virtual F* SASSY::cfi::PlugInFactoryT< F, P >::make ( ) [inline], [virtual]

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implements SASSY::cfi::PlugInFamilyFactoryT< F >.

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

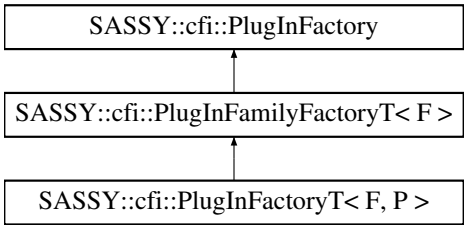
- plugin.h

7.47 SASSY::cfi::PlugInFamilyFactoryT< F > Class Template Reference

Template base class for families of plug-in factories.

`#include <cfi/plugin.h>`

Inheritance diagram for SASSY::cfi::PlugInFamilyFactoryT< F >:



Public Member Functions

- virtual F * make ()=0

  Construct a plug-in object.

7.47.1 Detailed Description

template<class F>class SASSY::cfi::PlugInFamilyFactoryT< F >

Template base class for families of plug-in factories.

Each family of plug-in objects will have its own API that is specified by the application and implemented by the plug-in library. This template allows the PlugInMgr to return an object of the correct type without having knowledge of that type built in.

7.47.2 Member Function Documentation

7.47.2.1 template<class F> virtual F* SASSY::cfi::PlugInFamilyFactoryT< F >::make ( ) [pure virtual]

Construct a plug-in object.

Returns

Pointer to new plug-in object

Implements SASSY::cfi::PlugInFactory.

Implemented in SASSY::cfi::PlugInFactoryT< F, P >.

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

- plugin.h

7.48 SASSY::cfi::PlugInLib Class Reference

Manages an instance of a dynamically loaded shared library.

`#include <cfi/plugin.h>`

Public Types

- enum Mode { AutoRun, OnDemand, StayResident }

  Control lifetime of the library.

Public Member Functions

- PlugInLib (const Path &libPath)

  Constructor.
- ~PlugInLib ()

  Destructor.
- void setMode (Mode m)

  Set the lifetime for a plug-in.
- Mode getMode ()

  Get the mode for the entire library.
- bool loaded ()

  Check if successfully loaded the library.
- void startUsage ()

  Indicate that the library is being used.
- void endUsage ()

  Advise that a plug-in is no longer using the library.
- bool inUse ()

  Test if the library is in use.
- const SASSY::Path & getPath ()

  Get the path to the library.

7.48.1 Detailed Description

Manages an instance of a dynamically loaded shared library.

This object is responsible for loading and unloading the library. It is also responsible for calling the initialisation and finalisation code within the library using the two extern "C" functions.

Note that a library may have more than one plug-in class defined, and these may even have different lifetimes.

7.48.2 Member Enumeration Documentation

7.48.2.1 enum SASSY::cfi::PlugInLib::Mode

Control lifetime of the library.

Enumerator

AutoRun The library can be removed after initialisation.

OnDemand The library can be removed once its unused.

StayResident The library is only removed on program termination.

7.48.3 Constructor & Destructor Documentation

7.48.3.1 PlugInLib::PlugInLib (const Path & *libPath*)

Constructor.

Parameters

libPath	The path to the shared library, from xini config

7.48.4 Member Function Documentation

7.48.4.1 Mode SASSY::cfi::PlugInLib::getMode ( ) [inline]

Get the mode for the entire library.

Returns

The lifetime mode.

7.48.4.2 const SASSY::Path& SASSY::cfi::PlugInLib::getPath ( ) [inline]

Get the path to the library.

Returns

the library path

7.48.4.3 bool SASSY::cfi::PlugInLib::inUse ( ) [inline]

Test if the library is in use.

Returns

True if a plug-in is still in active use.

7.48.4.4 bool PlugInLib::loaded ( )

Check if successfully loaded the library.

Returns

true if loaded OK

7.48.4.5 void PlugInLib::setMode (Mode *m*)

Set the lifetime for a plug-in.

This value is combined with the existing values to get a lifetime value for the overall library

Parameters

m	The mode for the plug-in

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

- plugin.h
- plugin.cpp

7.49 SASSY::cfi::PlugInMgr Class Reference

Manage the handling of plug-in shared libraries.

`#include <cfi/plugin.h>`

Public Member Functions

- void load (csr prog)

  Load the plug-ins listed in the xini config.
- void regn (const PlugInDetails &details)

  Allow a plug-in to register itself.
- void getDetails (std::vector< PlugInDescriptor > &plugins)

  Get list of plug ins.
- void release (PlugInLib *)

  Release a library.
- void shutdown ()

  Unload all the libraries prior to exiting the program.
- template<class PlugInFamily >
  std::unique_ptr< PlugInFamily > getPlugin (csr name)

  Get a plug-in.

Static Public Member Functions

- static PlugInMgr & instance ()

  Get a reference to the singleton object.

7.49.1 Detailed Description

Manage the handling of plug-in shared libraries.

7.49.2 Member Function Documentation

7.49.2.1 void PlugInMgr::load (csr *prog*)

Load the plug-ins listed in the xini config.

Parameters

prog	The name used to identify the set of plugins for this program.

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

- plugin.h
- plugin.cpp

7.50 rdf::Prefixes Class Reference

Manages the prefixes and namespaces for a World.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- Prefixes (World)

  Constructor.
- void base (URI uri)

  Set the base URI.
- void anonymous (URI uri)

  insert an anonymous namespace
- void base (const std::string &filename)

  Set the base URI.
- URI base () const

  Get the base URI.
- bool isBase (URI) const

  Check if a URI is the base URI.
- std::string removeBase (URI) const

  Strip the base URI from a URI.
- void insert (const std::string &prefix, URI)

  Save a prefix and its corresponding namespace URI.
- URI find (const std::string &)

  Find the namespace URI with the supplied prefix.
- std::string find (URI)

  Find the prefix for the supplied namespace URI.
- void update (const std::string &oldPrefix, const std::string &newPrefix, URI)

  Update a prefix and namespace.
- void remove (const std::string &)

  Remove a prefix and namespace.
- URI uriForm (const std::string &)

  Convert a prefix and fragment to a URI and fragment.
- std::string prefixForm (URI)

  Convert a URI and fragment to its prefix and fragment.
- std::map< std::string, URI >
  ::iterator begin ()

  Get an iterator for the saved prefixes.
- std::map< std::string, URI >
  ::iterator end ()

  get the end iterator for the saves prefixes.

7.50.1 Detailed Description

Manages the prefixes and namespaces for a World.

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

- rdfxx.h

7.51 SASSY::cfi::ProcessOwner Class Reference

Interface that allows the owner of a child process to be notified.

`#include <cfi/proc.h>`

Public Member Functions

- virtual ~ProcessOwner ()

  Destructor.
- virtual void processTerminated (AbstractChildProcess *cp)=0

  Notification that process terminated.

7.51.1 Detailed Description

Interface that allows the owner of a child process to be notified.

7.51.2 Member Function Documentation

7.51.2.1 virtual void SASSY::cfi::ProcessOwner::processTerminated (AbstractChildProcess * *cp*) [pure virtual]

Notification that process terminated.

Parameters

cp	The child process that ended.

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

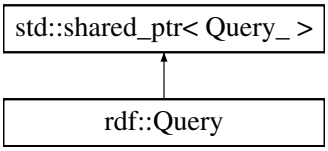
- proc.h

7.52 rdf::Query Class Reference

A shared pointer with constructors for the Query_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Query:



Public Member Functions

- Query (World, const std::string &query, const std::string &lang="sparql")

  Create a query.
- Query (World, const std::string &query, URI base_uri, const std::string &lang="sparql")

  Create a query.

7.52.1 Detailed Description

A shared pointer with constructors for the Query_ class.

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

- rdfxx.h

7.53 rdf::Query_ Class Reference

An abstract class defining the methods for an RDF Query.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Query_ ()

  Virtual destructor.
- virtual bool setLimit (int)=0

  Set the max number of results returned.
- virtual int getLimit () const =0

  Get the max number of results to be returned.
- virtual QueryResults execute (Model)=0

  Run the query on a model.

7.53.1 Detailed Description

An abstract class defining the methods for an RDF Query.

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

- rdfxx.h

7.54 rdf::QueryResult_ Class Reference

An abstract class defining the methods for an RDF Query Result.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual int count () const =0

  Get the number of bound values in the result.
- virtual std::string getBoundName (int offset) const =0

  Get the variable name at a position.
- virtual Node getBoundValue (int offset) const =0

  Get a value at a position.
- virtual Node getBoundValue (const std::string &name) const =0

  Get a value for a variable name.
- virtual std::string toString () const =0

  Convert all the names and values to a string.

7.54.1 Detailed Description

An abstract class defining the methods for an RDF Query Result.

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

- rdfxx.h

7.55 rdf::QueryResults_ Class Reference

An abstract class defining the methods for a set of Query Results.

`#include <rdfxx/rdfxx.h>`

Classes

- class iterator

  Provide a C++ iterator over the results of a RDF query.

Public Member Functions

- virtual ~QueryResults_ ()

  Virtual destructor.
- virtual bool success () const =0

  Check if the query got any results.
- virtual std::string toString ()=0

  Convert all results to a string. This consumes the results.
- virtual std::string toString (URI syntax, URI base)=0

  Convert the results to a string in the specified syntax.
- virtual iterator begin () const =0

  Get iterator at start of result set.
- virtual iterator end () const =0

  Get iterator past the end of result set.

7.55.1 Detailed Description

An abstract class defining the methods for a set of Query Results.

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

- rdfxx.h

7.56 rdf::QueryString Class Reference

A class for assistingin the preparation of a SPARQL query.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- QueryString ()

  Create empty query.
- void addPrefix (const std::string &id, const std::string &uri)

  Add a prefix and namespace to the query.
- void setVariables (const std::string &variables)

  Set the variables to search for.
- void addCondition (const std::string &condition)

  Add a condition to the query.
- void orderBy (const std::string &order)

Add an ordering clause to the query.

- operator std::string ()

  Get the query as a string.

7.56.1 Detailed Description

A class for assistingin the preparation of a SPARQL query.

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

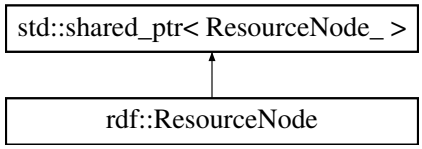
- rdfxx.h

7.57 rdf::ResourceNode Class Reference

A shared pointer with constructors for the ResourceNode_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::ResourceNode:



Public Member Functions

- ResourceNode (World, URI)

  Create resource node based on a URI.

- ResourceNode (World, Concept)

  Create resource node for a concept.

- ResourceNode (World, int)

  Create a numbered resource node for containers.

- ResourceNode (Node resourceNode)

  Convert a Node into a ResourceNode.

7.57.1 Detailed Description

A shared pointer with constructors for the ResourceNode_ class.

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

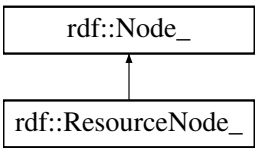
- rdfxx.h

7.58 rdf::ResourceNode_ Class Reference

An abstract class defining the methods for an RDF Resource Node.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::ResourceNode_:



Public Member Functions

- virtual std::string toString () const =0

  Get a string representation of the node.
- virtual std::string toString (const Format &) const =0

  Get a string representation of the node using the supplied formatting instructions.
- virtual URI toURI () const =0

  Get the URI for a Node.
- virtual int listItemOrdinal () const =0

  Get the ordinal value for a list item.

7.58.1 Detailed Description

An abstract class defining the methods for an RDF Resource Node.

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

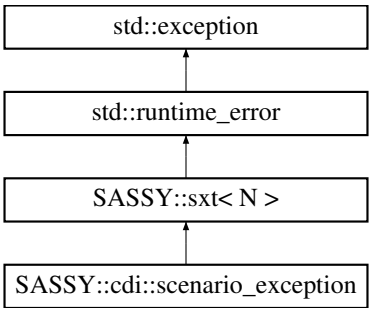
- rdfxx.h

7.59 SASSY::cdi::scenario_exception Class Reference

throw this to abandon an entire scenario

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::scenario_exception:



Public Member Functions

- scenario_exception ()

  Constructor.

Additional Inherited Members

7.59.1 Detailed Description

throw this to abandon an entire scenario

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

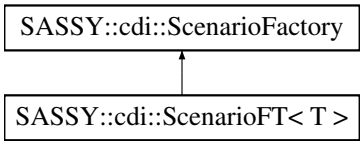
- testmgr.h

7.60 SASSY::cdi::ScenarioFactory Class Reference

Define a type for scenario factories.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::ScenarioFactory:



Public Member Functions

- ScenarioFactory ()
  Constructor.
- virtual ~ScenarioFactory ()
  Destructor.
- virtual AbstractScenario * make (csr name)=0
  Create an scenario object.

7.60.1 Detailed Description

Define a type for scenario factories.

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

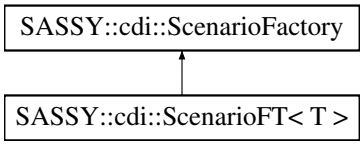
- testmgr.h

7.61 SASSY::cdi::ScenarioFT< T > Class Template Reference

Responsible for creating a scenario of some type.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::ScenarioFT< T >:



Public Member Functions

- AbstractScenario * make (csr name)

  Create a scenario object.

7.61.1 Detailed Description

template<class T>class SASSY::cdi::ScenarioFT< T >

Responsible for creating a scenario of some type.

7.61.2 Member Function Documentation

7.61.2.1 template<class T > AbstractScenario* SASSY::cdi::ScenarioFT< T >::make (csr *name*) [inline], [virtual]

Create a scenario object.

Returns

Pointer to a new scenario object.

Implements SASSY::cdi::ScenarioFactory.

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

- testmgr.h

7.62 SASSY::cdi::ScenarioResults Struct Reference

Responsible for storing the results of testing for a scenario.

`#include <cfi/testmgr.h>`

Public Member Functions

- ScenarioResults ()

  Constructor.

Public Attributes

- int mNumTests

  total number of tests

- int mNumErrors

  number of errors reported

- bool failure

  true if scenario tests failed

7.62.1 Detailed Description

Responsible for storing the results of testing for a scenario.

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

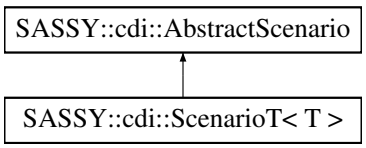
- testmgr.h

7.63 SASSY::cdi::ScenarioT< T > Class Template Reference

Responsible for installing a factory to create scenarios of the required type.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::ScenarioT< T >:



Public Member Functions

- ScenarioT (csr name)

  Constructor.

Static Public Member Functions

- static void install (csr nm)

  Install a scenario factor into the Tester.

Additional Inherited Members

7.63.1 Detailed Description

template<class T>class SASSY::cdi::ScenarioT< T >

Responsible for installing a factory to create scenarios of the required type.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 template<class T> SASSY::cdi::ScenarioT< T >::ScenarioT (csr *name*) [inline], [explicit]

Constructor.

Parameters

name	The name of the scenario.

7.63.3 Member Function Documentation

7.63.3.1 template<class T> static void SASSY::cdi::ScenarioT< T >::install (csr *nm*) [inline], [static]

Install a scenario factor into the Tester.

Parameters

nm	The name of the scenario

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

- testmgr.h

7.64 SASSY::cfi::Semaphore Class Reference

A semaphore for managing exclusive access to resources across multiple processes.

`#include <cfi/ipc.h>`

Public Member Functions

- Semaphore (char id)

  Constructor.
- ~Semaphore ()

  Destructor.
- void acquire ()

  wait until available
- void release ()

  release exclusive lock

7.64.1 Detailed Description

A semaphore for managing exclusive access to resources across multiple processes.

Provides the ability to limit access to a resource to one process at a time.

The semaphore uses a second semaphore to manage its lifetime. The first process to create it creates the system wide semaphore, and the last to delete it removes the system wide semaphore.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 Semaphore::Semaphore (char *id*)

Constructor.

Parameters

id	an identifier for the semaphore.

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

- ipc.h
- ipc.cpp

7.65 SASSY::cfi::SemaphoreLock Class Reference

Mutual exclusion lock.

`#include <cfi/ipc.h>`

Public Member Functions

- SemaphoreLock (Semaphore &s)

  Constructor which acquires a lock on the supplied semaphore.
- ~SemaphoreLock ()

  Destructor which releases the lock.

7.65.1 Detailed Description

Mutual exclusion lock.

Wraps the Semaphore so that it will be automatically released even if an exception is thrown.

7.65.2 Constructor & Destructor Documentation

7.65.2.1 SemaphoreLock::SemaphoreLock (Semaphore & *s*) [explicit]

Constructor which acquires a lock on the supplied semaphore.

Parameters

s	The semaphore being managed

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

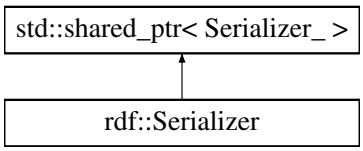
- ipc.h
- ipc.cpp

7.66 rdf::Serializer Class Reference

A shared pointer with constructors for the Serializer_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Serializer:



Public Member Functions

- Serializer ()

  Create a nullptr shared pointer.
- Serializer (Serializer_ *)

  replicate the shared pointer constructor
- Serializer (World, const std::string &name="rdfxml", const std::string &syntax_mime="")

  Create a serialiser using the specified syntax.
- Serializer (World, const std::string &name, URI syntax_uri)

  Create a serialiser using the specified syntax.

7.66.1 Detailed Description

A shared pointer with constructors for the Serializer_ class.

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

- rdfxx.h

7.67 rdf::Serializer_ Class Reference

An abstract class defining the methods for an RDF Serializer.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Serializer_ ()

  Virtual destructor.
- virtual bool setNamespace (URI, const std::string &prefix)=0

  Add a namespace and prefix to the output.
- virtual bool toFile (const std::string &filename, Model)=0

  Write the model to a file.
- virtual bool toFile (const std::string &filename, Model, URI base_uri)=0

  Write the model to a file.

7.67.1 Detailed Description

An abstract class defining the methods for an RDF Serializer.

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

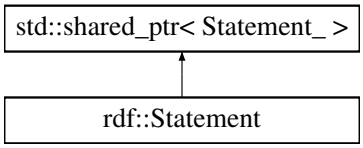
- rdfxx.h

7.68 rdf::Statement Class Reference

A shared pointer with constructors for the Statement_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Statement:



Public Member Functions

- Statement ()

  Create a nullptr based default statement.
- Statement (World)

Create an empty statement.

- Statement (World, Node subject, Node predicate, Node object)

  Create a statement with subject, predicate and object nodes.
- Statement (Statement_ *)

  Replicate the shared_ptr<> constructor.
- Statement (StatementRef)

  Convert a weak pointer to a shared pointer.

7.68.1 Detailed Description

A shared pointer with constructors for the Statement_ class.

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

- rdfxx.h

7.69 rdf::Statement_ Class Reference

An abstract class defining the methods for an RDF Statement.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Statement_ ()

  Virtual destructor.
- virtual Statement copy () const =0

  Clone the statement.
- virtual void subject (Node n)=0

  Set the subject node.
- virtual NodeRef subject () const =0

  Get a reference to the subject node.
- virtual void predicate (Node n)=0

  Set the predicate node.
- virtual NodeRef predicate () const =0

  Get a reference to the predicate node.
- virtual void object (Node n)=0

  Get the object node.
- virtual NodeRef object () const =0

  Get a reference to the object node.
- virtual bool isComplete () const =0

  Check if all three nodes are defined.
- virtual bool match (Statement) const =0

  Compare if the non-null nodes arethe same.
- virtual void clear ()=0

  Remove the nodes.
- virtual std::string toString () const =0

  Get a string representaion of the statement.
- virtual std::string toString (const Format &) const =0

  Get a string representaion of the statement using the specified formatting.
- virtual bool operator== (Statement) const =0

  Compare with another statement.

7.69.1 Detailed Description

An abstract class defining the methods for an RDF Statement.

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

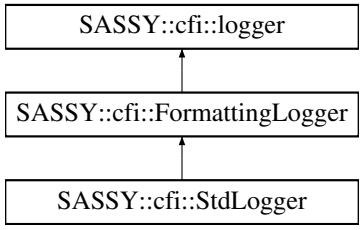
- rdfxx.h

7.70 SASSY::cfi::StdLogger Class Reference

Class for logging to the std output streams.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::StdLogger:



Public Types

- enum Stream { **COUT**, **CERR**, **CLOG** }

  Enumeration of the standard streams.

Public Member Functions

- StdLogger (Stream cc)

  Constructor.
- int log (Severity, csr line)

  log to the unnamed log
- int log (csr logName, Severity, csr line)

  log to the named log

Additional Inherited Members

7.70.1 Detailed Description

Class for logging to the std output streams.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 SASSY::cfi::StdLogger::StdLogger (Stream *cc*) [inline],[explicit]

Constructor.

Parameters

cc	enum to specify the stream to write to

7.70.3 Member Function Documentation

7.70.3.1 int StdLogger::log (Severity *sev*, csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.70.3.2 int StdLogger::log (csr *logName*, Severity *sev*, csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

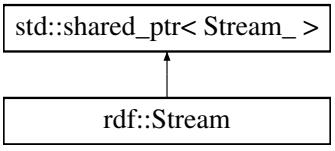
- log.h
- log.cpp

7.71 rdf::Stream Class Reference

A shared pointer with constructors for the Stream_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::Stream:



Public Member Functions

- Stream ()

  Create a nullptr shared pointer.
- Stream (World)

  Create a stream object.
- Stream (Stream_ *)

  Replicate the shared pointer constructot.
- Stream (World, Parser, URI, URI base)

  Create a stream from the specified parser.

7.71.1 Detailed Description

A shared pointer with constructors for the Stream_ class.

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

- rdfxx.h

7.72 rdf::Stream_ Class Reference

An abstract class defining the methods for an RDF Stream.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~Stream_ ()

  Virtual destructor.
- virtual bool end ()=0

  Check if at the end of the stream.
- virtual bool next ()=0

  Move to the next statement in the stream.
- virtual StatementRef current ()=0

  Get a reference to the current statement.

7.72.1 Detailed Description

An abstract class defining the methods for an RDF Stream.

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

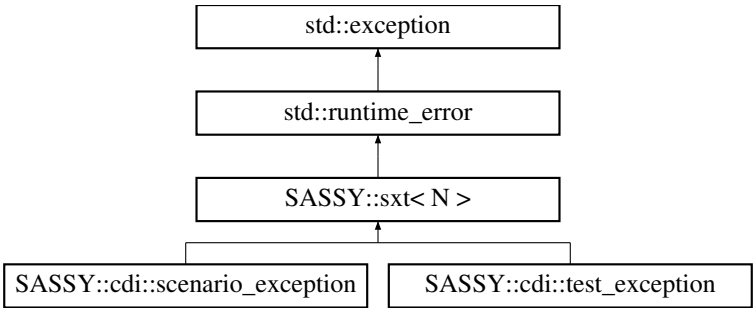
- rdfxx.h

7.73 SASSY::sxt< N > Class Template Reference

An exception object with severity levels.

`#include <cfi/sx.h>`

Inheritance diagram for SASSY::sxt< N >:



Public Member Functions

- sxt (Severity s)
  constructor
- sxt (Severity s, const std::string &f, int ln)
  constructor
- template<class T >
  sxt & operator<< (T x)
  Provide stream syntax for the exception object.
- Severity severity ()
  get the severity of the exception
- virtual const char * what () const throw ()
  get the message from the exception

Static Public Member Functions

- static const std::string & sevToString (SASSY::Severity s)
  Convert a severity into a string.

Protected Attributes

- Severity mSeverity
  severity level of the exception
- std::string buff
  message built into this
- std::string file
  source file from which it was thrown
- int line
  line number in the source file
- int ptr
  insertion point for message text

7.73.1 Detailed Description

template<typename N>class SASSY::sxt< N >

An exception object with severity levels.

The sxt class is an exception object that includes severity levels and has stream syntax for creating messages.

Typical usage: throw sxt(SASSY::Error) << "something went wrong: " << strerror(errno);

A macro version that automatically records the file name and line number is also defined. It would be used like:

throw SX(Warning) << "there is a problem";

7.73.2 Constructor & Destructor Documentation

7.73.2.1 template<typename N > SASSY::sxt< N >::sxt (Severity *s*) [inline], [explicit]

constructor

Parameters

s	The severity of the problem.

7.73.2.2 template<typename N > SASSY::sxt< N >::sxt (Severity *s*, const std::string & *f*, int *ln*) [inline]

constructor

Parameters

s	the severity of the exception
f	the file name of the source file
ln	the line number of the source file

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

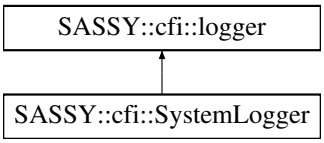
- sx.h

7.74 SASSY::cfi::SystemLogger Class Reference

logger that interfaces to the Linux syslog

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::SystemLogger:



Public Member Functions

- SystemLogger (csr ident)
- int log (Severity, csr line)

  log to the unnamed log
- int log (csr logName, Severity, csr line)

  log to the named log

Additional Inherited Members

7.74.1 Detailed Description

logger that interfaces to the Linux syslog

7.74.2 Constructor & Destructor Documentation

7.74.2.1 SystemLogger::SystemLogger (csr *ident*)

Parameters

ident	the identifier for the log.

7.74.3 Member Function Documentation

7.74.3.1 int SystemLogger::log (Severity *sev*, csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.74.3.2 int SystemLogger::log (csr *logName*, Severity *sev*, csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

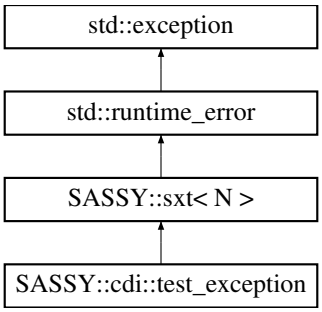
- log.h
- log.cpp

7.75 SASSY::cdi::test_exception Class Reference

throw this to abandon a particular test case

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::test_exception:



Public Member Functions

- test_exception ()

  Constructor.

Additional Inherited Members

7.75.1 Detailed Description

throw this to abandon a particular test case

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

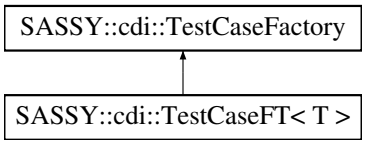
- testmgr.h

7.76 SASSY::cdi::TestCaseFactory Class Reference

Define a base type for test case factories.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestCaseFactory:



Public Member Functions

- virtual ~TestCaseFactory ()

  Destructor.
- virtual AbstractTestCase * make (csr nm)=0

  Create an instance of the test case object.

7.76.1 Detailed Description

Define a base type for test case factories.

7.76.2 Member Function Documentation

7.76.2.1 virtual AbstractTestCase* SASSY::cdi::TestCaseFactory::make (csr *nm*) [pure virtual]

Create an instance of the test case object.

Parameters

nm	The name of the test case

Returns

A pointer to a new instance of the test case.

Implemented in SASSY::cdi::TestCaseFT< T >.

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

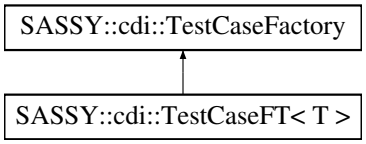
- testmgr.h

7.77 SASSY::cdi::TestCaseFT< T > Class Template Reference

Responsible for creating a test case of some type.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestCaseFT< T >:



Public Member Functions

- AbstractTestCase * make (csr nm)

  Create a test case object.

7.77.1 Detailed Description

template<class T>class SASSY::cdi::TestCaseFT< T >

Responsible for creating a test case of some type.

7.77.2 Member Function Documentation

7.77.2.1 template<class T > AbstractTestCase* SASSY::cdi::TestCaseFT< T >::make (csr *nm*) [inline], [virtual]

Create a test case object.

Parameters

nm	The name of the test case

Returns

A pointer to a new instance of the test case.

Implements SASSY::cdi::TestCaseFactory.

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

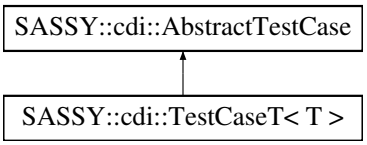
- testmgr.h

7.78 SASSY::cdi::TestCaseT< T > Class Template Reference

Responsible for installing a factory for the test case.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestCaseT< T >:



Public Member Functions

- TestCaseT (csr nm)
  Constructor.

Static Public Member Functions

- static void install (csr nm, csr scn)
  Install a factory for the test case.

Additional Inherited Members

7.78.1 Detailed Description

template<class T>class SASSY::cdi::TestCaseT< T >

Responsible for installing a factory for the test case.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 template<class T > SASSY::cdi::TestCaseT< T >::TestCaseT (csr *nm*) [inline]

Constructor.

Parameters

nm	The name of the test case.

7.78.3 Member Function Documentation

7.78.3.1 template<class T > static void SASSY::cdi::TestCaseT< T >::install (csr *nm*, csr *scn*) [inline], [static]

Install a factory for the test case.

Parameters

nm	The name of the test case
scn	The name of enclosing scenario.

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

- testmgr.h

7.79 SASSY::cdi::Tester Class Reference

Responsible for managing the testing.

`#include <cfi/testmgr.h>`

Public Member Functions

- void configure (csr testName)
  specify the name of the test as in the configuration file
- csr getTestName ()
  Get the name of the test.
- virtual ~Tester ()
  Destructor.
- cfi::logstream & log ()
  Get a reference to the logging stream used for the testing.
- void addTestCase (csr name, csr scenario, TestCaseFactory *tcf)
  Add a test case factory.
- void runTestCase (csr scenarioName, csr testCaseName, ScenarioResults *)
  Run a specific test case.
- void getTestCases (csr scenarioName, std::set< std::string > &testCases)
  Get the test case names for a scenario.
- void addScenario (csr scenario, ScenarioFactory *sf)
  Add a scenario factory.
- void setTest (TestFactory *t)
  Set the main test object factory.
- AbstractTest * getTest ()
  Get the test object.
- AbstractScenario * getScenario ()
  Get the current scenario object.
- virtual void runTests ()
  Run all the tests.
- bool summary ()
  Print a summary of the testing to the log stream.

Static Public Member Functions

- static Tester & instance ()

  Get a reference to the Tester.

Protected Member Functions

- Tester ()

  Protected Constructor.
- void title ()

  Print title for the test.
- void installDefaults ()

  Install default test and scenario objects.
- void testScenario (csr scenario, ScenarioFactory *scn)

  Do testing of a scenario.

Protected Attributes

- std::string nameOfTest

  Identifier for test in the configuration file.
- std::string logName

  Identifier for the log.
- bool failed

  Flag to hold overall testing result.
- std::map< std::string, std::map< std::string, TestCaseFactory * > > tests

  Factories for test cases indexed by name and scenario name.
- std::map< std::string, ScenarioFactory * > scenarios

  Factories for scenarios indexed by scenario name.
- std::map< std::string, ScenarioResults * > results

  Results for scenario tests indexed by scenario name.
- TestFactory * theTest

  Factor for the overall test.
- AbstractTest * absTest

  Pointer to the abstract test object. Only valid during the test run.
- AbstractScenario * absScenario

  Pointer to the abstract scenario object. Only valid during the test run.

7.79.1 Detailed Description

Responsible for managing the testing.

A singleton that manages the testing, accumlates the test results and provides a report.

7.79.2 Member Function Documentation

7.79.2.1 void Tester::addScenario (csr *scenario*, ScenarioFactory * *sf*)

Add a scenario factory.

Parameters

scenario	The name of the scenario
sf	The factory for the scenario

7.79.2.2 void Tester::addTestCase (csr *name*, csr *scenario*, TestCaseFactory * *tcf*)

Add a test case factory.

Parameters

name	The name of the test case.
scenario	The name of the enclosing scenario.
tcf	The factory for the test case.

7.79.2.3 void Tester::configure (csr *testName*)

specify the name of the test as in the configuration file

Parameters

testName	The name of the test in the config file.

7.79.2.4 AbstractScenario* SASSY::cdi::Tester::getScenario ( ) [inline]

Get the current scenario object.

Returns

The current scenario object

7.79.2.5 AbstractTest* SASSY::cdi::Tester::getTest ( ) [inline]

Get the test object.

Returns

The test object

7.79.2.6 csr SASSY::cdi::Tester::getTestName ( ) [inline]

Get the name of the test.

Returns

the name of test

7.79.2.7 Tester & Tester::instance ( ) [static]

Get a reference to the Tester.

Returns

A reference to the Tester singleton object.

7.79.2.8 logstream & Tester::log ( )

Get a reference to the logging stream used for the testing.

Returns

A reference to the logging stream

7.79.2.9 void SASSY::cdi::Tester::setTest (TestFactory * t) [inline]

Set the main test object factory.

Parameters

t	The factory for the test object.

7.79.2.10 bool Tester::summary ( )

Print a summary of the testing to the log stream.

Returns

true if the testing found no unexpected errors

7.79.2.11 void Tester::testScenario (csr *scenario,* ScenarioFactory * *scn*) [protected]

Do testing of a scenario.

Parameters

scenario	The name of the scenario.
scn	The factory for creating the scenario object

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

- testmgr.h
- testmgr.cpp

7.80 SASSY::cdi::TestEvent Class Reference

Represent an event in the object under test.

`#include <cfi/testmgr.h>`

Public Member Functions

- TestEvent (const std::string &s)

  Constructor.
- const std::string & id ()

  Get the identity of the event.

7.80.1 Detailed Description

Represent an event in the object under test.

These objects are placed into a queue when an event of interest occurs in the object under test. The AsyncTestCase object will get the object from the queue and perform checks on the object under test.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 SASSY::cdi::TestEvent::TestEvent (const std::string & *s*) [inline]

Constructor.

Parameters

s	The identity of the event.

7.80.3 Member Function Documentation

7.80.3.1 const std::string& SASSY::cdi::TestEvent::id ( ) [inline]

Get the identity of the event.

Returns

The identifier for the event.

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

- testmgr.h

7.81 SASSY::cdi::TestEventQueue Class Reference

Responsible for queuing TestEvents.

`#include <cfi/testmgr.h>`

Public Member Functions

- ~TestEventQueue ()

  Destructor.
- void enqueueEvent (TestEvent *)

  Place the event on the queue.

Static Public Member Functions

- static TestEventQueue & instance ()

  Get reference to the singleton queue object.
- static void event (csr id)

  Create a TestEvent, queue it, and wait for it to be processed.

Public Attributes

- std::mutex eventMx

  Control access to the queue.
- std::list< TestEvent * > events

  The queue.
- std::set< std::string > runningEvents

  The set of events being handled.
- std::mutex functionsMx

  Control access to list of running events.
- std::condition_variable eventCV

  Wait for event to be queued or processed.

Protected Member Functions

- TestEventQueue ()

  Constructor.

7.81.1 Detailed Description

Responsible for queuing TestEvents.

The queue provides a condition variable that allows the test case to wait until an event has been queued.

7.81.2 Member Function Documentation

7.81.2.1 void TestEventQueue::enqueueEvent (TestEvent * *ev*)

Place the event on the queue.

Parameters

ev	The TestEvent object to place on the queue

7.81.2.2 void TestEventQueue::event (csr *id*) [static]

Create a TestEvent, queue it, and wait for it to be processed.

Parameters

id	The identity for the event

7.81.2.3 TestEventQueue & TestEventQueue::instance ( ) [static]

Get reference to the singleton queue object.

Returns

reference to the TestEventQueue

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

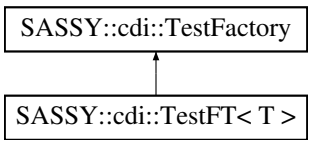
- testmgr.h
- testmgr.cpp

7.82 SASSY::cdi::TestFactory Class Reference

Responsible for creating an object that manages the resources for the entire test.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestFactory:



Public Member Functions

- virtual ~TestFactory ()

  Destructor.
- virtual AbstractTest * make ()=0

  Create an instance of the test object.

7.82.1 Detailed Description

Responsible for creating an object that manages the resources for the entire test.

7.82.2 Member Function Documentation

7.82.2.1 virtual AbstractTest* SASSY::cdi::TestFactory::make ( ) [pure virtual]

Create an instance of the test object.

Returns

Pointer to the test object.

Implemented in SASSY::cdi::TestFT< T >.

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

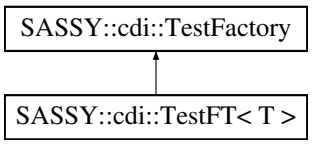
- testmgr.h

7.83 SASSY::cdi::TestFT< T > Class Template Reference

Responsible for creating a test object of the required type.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestFT< T >:



Public Member Functions

- AbstractTest * make ()
  Construct a Test object.

7.83.1 Detailed Description

template<class T>class SASSY::cdi::TestFT< T >

Responsible for creating a test object of the required type.

7.83.2 Member Function Documentation

7.83.2.1 template<class T > AbstractTest* SASSY::cdi::TestFT< T >::make ( ) [inline], [virtual]

Construct a Test object.

Returns

pointer to a new object derived from AbstractTest

Implements SASSY::cdi::TestFactory.

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

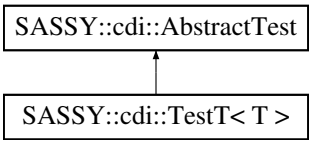
- testmgr.h

7.84 SASSY::cdi::TestT< T > Class Template Reference

Responsible for installing a factory for making test objects.

`#include <cfi/testmgr.h>`

Inheritance diagram for SASSY::cdi::TestT< T >:



Static Public Member Functions

- static void install ()
  Install a factory for a type of AbstractTest object.

Additional Inherited Members

7.84.1 Detailed Description

template<class T>class SASSY::cdi::TestT< T >

Responsible for installing a factory for making test objects.

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

- testmgr.h

7.85 SASSY::cdi::Trace Class Reference

This class is used to create trace log entries.

`#include <cfi/trace.h>`

Public Member Functions

- Trace (int n)

  Constructor.
- Trace (int n, csr file)

  Constructor.
- template<class T >
  Trace & operator<< (T x)

  Stream output operator.
- ~Trace ()

  Destructor.

7.85.1 Detailed Description

This class is used to create trace log entries.

Objects of this class should only be transient. i.e. don't actually name the object, so that when the statement ends the destructor is called immediately.

Typical usage is TRACE(3) << "my trace message";

7.85.2 Constructor & Destructor Documentation

7.85.2.1 Trace::Trace (int *n*) [explicit]

Constructor.

Parameters

n	debug level. See Tracer for detailed explanation.

7.85.2.2 Trace::Trace (int *n*, csr *file*)

Constructor.

Parameters

n	debug level. See Tracer for detailed explanation.
file	source file.

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

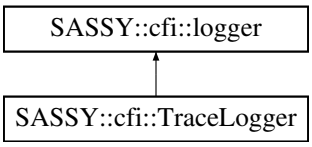
- trace.h
- trace.cpp

7.86 SASSY::cfi::TraceLogger Class Reference

Class for logging tracing output.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::TraceLogger:



Public Member Functions

- TraceLogger (const Path &fname)
- int log (Severity, csr line)

  log to the unnamed log
- int log (csr logName, Severity, csr line)

  log to the named log

Protected Attributes

- std::ofstream f

  The logging stream.

Additional Inherited Members

7.86.1 Detailed Description

Class for logging tracing output.

7.86.2 Constructor & Destructor Documentation

7.86.2.1 TraceLogger::TraceLogger (const Path & *fname*)

Parameters

fname	the name of the log file.

7.86.3 Member Function Documentation

7.86.3.1 int TraceLogger::log (Severity *sev,* csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.86.3.2 int TraceLogger::log (csr *logName,* Severity *sev,* csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

- log.h
- log.cpp

7.87 SASSY::cdi::Tracer Class Reference

This class manages the tracing for an application.

`#include <cfi/trace.h>`

Public Types

- enum CallEntry { **ENTER** =+1, **EXIT** =-1, **NONE** =0 }

  Indicator for entry or exit of a function.

Public Member Functions

- void log (csr file, int level, CallEntry call, csr text)

  Write a trace line to the log stream.

Static Public Member Functions

- static Tracer & instance ()

  Get an instance of the singleton object.

7.87.1 Detailed Description

This class manages the tracing for an application.

This class is responsible for managing tracing for an application.

It handles both normal trace messages and function call tracing.

Trace messages are sent to the log stream defined for tracing in the XINI configuration file. The messages have microsecond resolution timestamps to enable accurate comparison of messages from multiple processes.

Tracing can be controlled on a file by file basis by setting the debug level for the file in the tracing section of the XINI configuration file.

7.87.2 Member Function Documentation

7.87.2.1 void Tracer::log (csr *file*, int *level*, CallEntry *call*, csr *text*)

Write a trace line to the log stream.

Parameters

file	determines the debugging level for that file from sourceMap
level	the current debug level
call	+1 on entry, -1 on exit, 0 otherwise
text	the log message

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

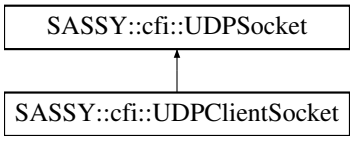
- trace.h
- trace.cpp

7.88 SASSY::cfi::UDPClientSocket Class Reference

A UDP client socket.

`#include <sos/udpsocket.h>`

Inheritance diagram for SASSY::cfi::UDPClientSocket:



Public Member Functions

- UDPClientSocket (csr host, int portid)
  Constructor.

Additional Inherited Members

7.88.1 Detailed Description

A UDP client socket.

A UDP socket which send messages to a specified server.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 UDPClientSocket::UDPClientSocket (csr *host,* int *portid*)

Constructor.

Parameters

host	the name of the server to send to
portid	the port for the listening server

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

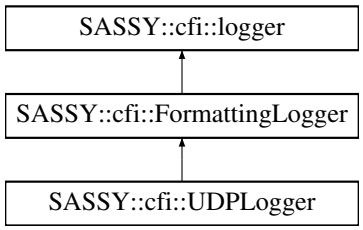
- udpsocket.h
- udpsocket.cpp

7.89 SASSY::cfi::UDPLogger Class Reference

Class for logging to a logging server.

`#include <cfi/log.h>`

Inheritance diagram for SASSY::cfi::UDPLogger:



Public Member Functions

- UDPLogger (csr hostname, int port)
- int log (Severity, csr line)

  log to the unnamed log
- int log (csr logName, Severity, csr line)

  log to the named log

Protected Attributes

- UDPClientSocket socket

  the udp socket connected to the logging server

Additional Inherited Members

7.89.1 Detailed Description

Class for logging to a logging server.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 UDPLogger::UDPLogger (csr *hostname,* int *port*)

Parameters

hostname	the identity of the log server in the configuration file
port	the port to send to

7.89.3 Member Function Documentation

7.89.3.1 int UDPLogger::log (Severity *sev*, csr *line*) [virtual]

log to the unnamed log

Parameters

sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

7.89.3.2 int UDPLogger::log (csr *logName*, Severity *sev*, csr *line*) [virtual]

log to the named log

Parameters

logName	the name of the log stream to write to
sev	the severity level for the message
line	the message to place in the log

Returns

the number of characters from the string that were written.

Implements SASSY::cfi::logger.

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

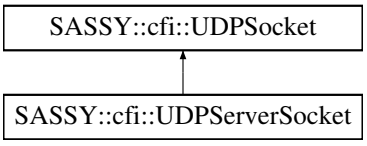
- log.h
- log.cpp

7.90 SASSY::cfi::UDPServerSocket Class Reference

A UDP Server socket.

`#include <udpsocket.h>`

Inheritance diagram for SASSY::cfi::UDPServerSocket:



Public Member Functions

- UDPServerSocket (int portid)

  Constructor.
- void send (csr message, csr host, int port)

  send message to specified host and port. Used for replies.

Additional Inherited Members

7.90.1 Detailed Description

A UDP Server socket.

A UDP socket which accepts incoming packets.

7.90.2 Constructor & Destructor Documentation

7.90.2.1 UDPServerSocket::UDPServerSocket (int *portid*)

Constructor.

Parameters

portid	the port to listen on.

7.90.3 Member Function Documentation

7.90.3.1 void UDPServerSocket::send (csr *message,* csr *host,* int *port*)

send message to specified host and port. Used for replies.

Parameters

message	the text to send.
host	the machine to send to.
port	the port to send to.

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

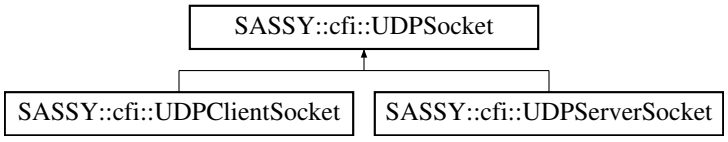
- udpsocket.h
- udpsocket.cpp

7.91 SASSY::cfi::UDPSocket Class Reference

An abstract base class with common stuff for UDP sockets.

`#include <sos/udpsocket.h>`

Inheritance diagram for SASSY::cfi::UDPSocket:



Public Member Functions

- virtual ~UDPSocket ()
  destructor
- int fd ()
  get the file descriptor for the socket
- void send (csr message)
  send a message via the socket's file descriptor
- void recv (std::string &message, std::string &host, int &port)
  wait for the next incoming message

Static Public Attributes

- static const int BUFFER_SIZE = 16000

Protected Member Functions

- UDPSocket ()
  constructor

Protected Attributes

- struct sockaddr_in addr
  internal address structure
- unsigned int addr_len
  size of address structure
- int sd
  file descriptor

7.91.1 Detailed Description

An abstract base class with common stuff for UDP sockets.

The common code for client and server udp sockets which manages the file descriptor and the buffer for the messages.

7.91.2 Member Function Documentation

7.91.2.1 void UDPSocket::recv (std::string & *message*, std::string & *host*, int & *port*)

wait for the next incoming message

Parameters

message	a string into which the message is placed
host	the host name of the sending machine
port	the port of the sending machine.

7.91.2.2 void UDPSocket::send (csr *message*)

send a message via the socket's file descriptor

Parameters

message	the text to send.

7.91.3 Member Data Documentation

7.91.3.1 const int SASSY::cfi::UDPSocket::BUFFER_SIZE = 16000 [static]

the max size for udp packets is 64K but we should keep them as small as possible and send several if needs be.

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

- udpsocket.h
- udpsocket.cpp

7.92 rdf::Universe Class Reference

A singleton class responsible for managing the World objects.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- World world (const std::string &name)
  Create, or return an existing world object.

Static Public Member Functions

- static Universe & instance ()
  Get a reference to the universe object.

7.92.1 Detailed Description

A singleton class responsible for managing the World objects.

The Universe object allows the using program to access a World object by name. Access will be protected by a mutex so that different threads can each use RDF processing.

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

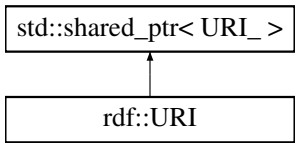
- rdfxx.h

7.93 rdf::URI Class Reference

A shared pointer with constructors for the URI_ class.

`#include <rdfxx/rdfxx.h>`

Inheritance diagram for rdf::URI:



Public Member Functions

- URI ()

  Default constructor.
- URI (URI_ *)

  Create shared pointer to new URI_ object.
- URI (World, const std::string &uri_string)

  Create URI using a string.
- URI (World, Concept)

  Create URI using a Concept.
- URI (const std::string &filename, World)

  Convert a file name into a URI.
- URI (const std::string &uri_string, URI source_uri, URI base_uri)

  Create URI the has source_uri replaced by base_uri in uri_string.
- URI (URI base_uri, const std::string &uri_string)

  Create URI where the uri_string replaces anything after the final '/' in base_uri.
- URI (const std::string &local_name, URI basr_uri)

  Create a URI from base URI and local name.

7.93.1 Detailed Description

A shared pointer with constructors for the URI_ class.

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

- rdfxx.h

7.94 rdf::URI_ Class Reference

An abstract class defining the methods for an RDF URI.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~URI_ ()

  Virtual destructor.
- virtual URI copy () const =0

  Clonethe URI.
- virtual URI trim (World) const =0

  Strip off fragment.
- virtual std::string toString () const =0

  Get a string representation.
- virtual bool operator== (URI) const =0

  Compare with another URI.
- virtual bool isFileName () const =0

  Check if URI represents a file path.
- virtual std::string toFileName () const =0

  Convert the URI to a file path name.

7.94.1 Detailed Description

An abstract class defining the methods for an RDF URI.

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

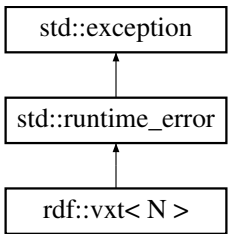
- rdfxx.h

7.95 rdf::vxt< N > Class Template Reference

An exception object with severity levels.

`#include <rdfxx/except.h>`

Inheritance diagram for rdf::vxt< N >:



Public Member Functions

- vxt (Severity s)
  constructor
- vxt (Severity s, const std::string &f, int ln)
  constructor
- template<class T >
  vxt & operator<< (T x)
  Provide stream syntax for the exception object.
- Severity severity ()
  get the severity of the exception
- virtual const char * what () const throw ()
  get the message from the exception

Static Public Member Functions

- static const std::string & sevToString (Severity s)
  Convert a severity into a string.

Protected Attributes

- Severity mSeverity
  severity level of the exception
- std::string buff
  message built into this
- std::string file
  source file from which it was thrown
- int line

line number in the source file

- int ptr

  insertion point for message text

7.95.1 Detailed Description

template<typename N>class rdf::vxt< N >

An exception object with severity levels.

The vxt class is an exception object that includes severity levels and has stream syntax for creating messages.

Typical usage: throw vxt(VICI::Error) << "something went wrong: " << strerror(errno);

A macro version that automatically records the file name and line number is also defined. It would be used like:

throw VX(Warning) << "there is a problem";

7.95.2 Constructor & Destructor Documentation

7.95.2.1 template<typename N > rdf::vxt< N >::vxt (Severity *s*) [inline], [explicit]

constructor

Parameters

s	The severity of the problem.

7.95.2.2 template<typename N > rdf::vxt< N >::vxt (Severity *s*, const std::string & *f*, int *ln*) [inline]

constructor

Parameters

s	the severity of the exception
f	the file name of the source file
ln	the line number of the source file

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

- except.h

7.96 rdf::World_ Class Reference

An abstract class defining the methods for an RDF World.

`#include <rdfxx/rdfxx.h>`

Public Member Functions

- virtual ~World_ ()

  Virtual destructor.

- virtual void registerErrorClient (ErrorClient *, bool warnings, bool errors)=0

  Register a client object to receive error and/or error messages.

- virtual void deregisterErrorClient (ErrorClient *)=0

  Remove a client that was to get error messages.

- virtual Serializer defaultSerializer ()=0

  Get a default serialiser - depreicated.
- virtual Prefixes & prefixes ()=0

  Get a reference to the saved prefixes.

7.96.1 Detailed Description

An abstract class defining the methods for an RDF World.

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

- rdfxx.h

7.97 SASSY::cfi::XINI Class Reference

Load an XML configuration file.

`#include <cfi/xini.h>`

Inheritance diagram for SASSY::cfi::XINI:

Public Member Functions

- bool getVal (const std::string &xpath_expression, std::string &val)

  get a value from the config file.
- int getVals (const std::string &xpath_expr, std::vector< std::string > &results)

  get a set of values
- void getPath (const std::string &xpath_expr, SASSY::Path &path)

  get a path from a set of paths
- const Path & getConfigFilename () const

  get the name of the config file being used

Static Public Member Functions

- static void configure (const Path &p)

  Configure XINI to use the specified path.
- static void configure (int c, char **v)

  Configure XINI to use the command line args.
- static XINI & instance ()

  get a reference to the single instance of the class.

Protected Member Functions

- XINI ()

  constructor

Static Protected Attributes

- static int argc = 0

  number of command line args
- static char ** argv = NULL

  command line args
- static Path configFile

  path to config file
- static const std::string config

  configuration string defined by the using application using the format specified.

Friends

- class **::XiniTestCase**

7.97.1 Detailed Description

Load an XML configuration file.

Ini file, XML format, loaded by looking in the following locations:

- P = path provided on the command line.
- E = path provided in the environment
- C = the current working directory
- H = the user's home directory
- D = directories as listed

Containing application must provide a string containing the keys for the above search locations:

```
const string ini::config = "PECHD:-i:QM_CONF:qmd.xml:.qmd.xml:/etc/qmd/qmd.xml";
```

The above will first look for a command line parameter following -i, then it will look in the environment for a variable QM_CONF, then in the current directory for a file called qmd.xml, then for a file $HOME/.qmd.xml, and finally for /etc/qmd/qmd.xml

Any further paths added at the end are checked in sequence.

7.97.2 Member Function Documentation

7.97.2.1 void XINI::configure (const Path & *p*) [static]

Configure XINI to use the specified path.

Parameters

p	The path to the XML config file to use

7.97.2.2 void XINI::configure (int *c*, char ** *v*) [static]

Configure XINI to use the command line args.

Parameters

c	The number of command line args
v	the command line args

7.97.2.3 const Path & XINI::getConfigFilename ( ) const

get the name of the config file being used

Returns

path of the configuration file that was found.

7.97.2.4 void XINI::getPath (const std::string & *xpath_expr*, SASSY::Path & *path*)

get a path from a set of paths

Parameters

xpath_expr	An expression returning a set of paths to search
path	The first path that points to an existing file

7.97.2.5 bool XINI::getVal (const std::string & *xpath_expression*, std::string & *val*)

get a value from the config file.

Parameters

xpath_expression	an expression to find in the config file.
val	value is returned into this

Returns

true if the value exists, and is unique

7.97.2.6 int XINI::getVals (const std::string & *xpath_expr*, std::vector< std::string > & *results*)

get a set of values

Parameters

xpath_expr	an expression to find in the config file.
results	set of values returned in this

Returns

the size of the set

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

- xini.h
- xini.cpp

7.98 SASSY::cfi::Xml Class Reference

A C++ wrapper for libxml2.

`#include <cfi/xml.h>`

Inheritance diagram for SASSY::cfi::Xml:

Public Member Functions

- virtual ~Xml ()

  Destructor.
- void setDtd (csr name, const Path &dtd)

  Set DTD identifier to dtd, which is just the dtd file name.
- void getDtdIdentifiers (csr dtdName, std::string &publicId, std::string &systemId)

  Get the public identifier for the DTD.
- void setCompression (int rate)

  Set the compression rate for saving the document.
- void save ()

  Save the document.
- void save (const Path &fname)

  Save the document to a new file.
- bool dirty ()

  Check if the document has been modified.
- void setDirty (bool dt)

  Access to dirty flag if we change document outside of this interface.
- bool safeToSave ()

  Check if can save without clobbering other input.
- void reload ()

  Reload the XML file.

Protected Member Functions

- void createXPathContext ()

  Call this whenever mDoc is explicitly updated.
- void registerNamespace (const std::string &prefix, csr uri)
- Xml ()

  Constructor.
- void open (const Path &fname)

  Open an existing XML file.
- void create (const Path &fname, csr root_element)

  Create a new xml file and document.
- void freeDoc ()

  release the document from memory
- void setProp (xmlNodePtr node, csr prop, csr val)

  Set a string property.

- void setProp (xmlNodePtr node, csr prop, int val)

  Set an integer property.
- void setProp (xmlNodePtr node, csr prop, double val)

  Set a double property.
- void setContent (xmlNodePtr node, csr text)

  Set the content for the node.
- void setCDATAContent (xmlNodePtr node, csr text)

  Set the content for the node using CDATA.
- void getPropString (xmlNodePtr node, csr prop, std::string &val)

  Get the property value as a string.
- void getPropInt (xmlNodePtr node, csr prop, int &val)

  Get the property value as an integer.
- void getPropShort (xmlNodePtr node, csr prop, short &val)

  Get the property value as a short integer.
- void getPropDouble (xmlNodePtr node, csr prop, double &val)

  Get the property value as a double.
- std::string getNodeContent (xmlNodePtr p, int expand=1) const

  Get the content of the node.
- std::string getNodeName (xmlNodePtr p) const

  Get the name of the node.
- xmlNodePtr getRoot ()

  Get the root node.
- xmlNodePtr getChild (xmlNodePtr node, csr name) const

  Get a child node by name.
- int getChildren (xmlNodePtr node, std::vector< xmlNodePtr > &nodes) const

  Get the children of the node.
- xmlNodePtr newNode (xmlNodePtr node, csr name)

  Create a new node.
- void deleteNode (xmlNodePtr node)

  Delete a node and all of its children.
- int find (csr xpath_expression, std::vector< xmlNodePtr > &nodes)

  Find nodes matching an XPath.

Static Protected Member Functions

- static xmlNodePtr newTextChild (xmlNodePtr node, csr name, csr text)

  Create a new node with text content.

Protected Attributes

- Path mFilename

  the XML file that the object represents
- struct stat xmlStat

  time the file was last accessed by us
- xmlDocPtr mDoc

  pointer to the root document object
- xmlXPathContextPtr mCtx

  XML XPath context.
- bool mDirty

  true when an update has been made, and saving is required.

- bool mIsOpen

  true if the file is open
- int mCompression

  0 is uncompressed, thru 9 for max zlib compression

Static Protected Attributes

- static const int UMASK_RW_RW_R = 0664 ^ 0777

  Default umask for xml files created.
- static bool xpathinit = false

  flag to indicate if xpath has been initialized

7.98.1 Detailed Description

A C++ wrapper for libxml2.

The Xml class provides an object oriented wrapper for the libxml2 library. Applications will subclass this class to provide application specific functions for accessing the XML data.

7.98.2 Member Function Documentation

7.98.2.1 void Xml::create (const Path & *fname*, csr *root_element*) [protected]

Create a new xml file and document.

Parameters

fname	the name of the file which will be created on save.
root_element	the name of the root element

7.98.2.2 void Xml::deleteNode (xmlNodePtr *node*) [protected]

Delete a node and all of its children.

Parameters

node	the document node

7.98.2.3 bool SASSY::cfi::Xml::dirty ( ) [inline]

Check if the document has been modified.

Returns

true if the document has been modified.

7.98.2.4 int Xml::find (csr *xpath_expression*, std::vector< xmlNodePtr > & *nodes*) [protected]

Find nodes matching an XPath.

Parameters

xpath_expression	the expression to search for.
nodes	a vector for the matching nodes.

7.98.2.5 xmlNodePtr Xml::getChild (xmlNodePtr *node*, csr *name*) const [protected]

Get a child node by name.

Parameters

node	the document node
name	the name of the node.

7.98.2.6 int Xml::getChildren (xmlNodePtr *node*, std::vector< xmlNodePtr > & *nodes*) const [protected]

Get the children of the node.

Parameters

node	the document node
nodes	a vector for the children nodes.

7.98.2.7 void Xml::getDtdIdentifiers (csr *dtdName*, std::string & *publicId*, std::string & *systemId*)

Get the public identifier for the DTD.

Parameters

dtdName	the name for the DTD entry
publicId	the returned public identifier for the DTD
systemId	the returned system identifier for the DTD

7.98.2.8 string Xml::getNodeContent (xmlNodePtr *p*, int *expand* = 1) const [protected]

Get the content of the node.

Parameters

p	the document node
expand	if 1 then get the content of child nodes.

7.98.2.9 string Xml::getNodeName (xmlNodePtr *p*) const [protected]

Get the name of the node.

Parameters

p	the document node

7.98.2.10 void Xml::getPropDouble (xmlNodePtr *node*, csr *prop*, double & *val*) [protected]

Get the property value as a double.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.11 void Xml::getPropInt (xmlNodePtr *node*, csr *prop*, int & *val*) [protected]

Get the property value as an integer.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.12 void Xml::getPropShort (xmlNodePtr *node*, csr *prop*, short & *val*) [protected]

Get the property value as a short integer.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.13 void Xml::getPropString (xmlNodePtr *node*, csr *prop*, std::string & *val*) [protected]

Get the property value as a string.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.14 xmlNodePtr Xml::newNode (xmlNodePtr *node*, csr *name*) [protected]

Create a new node.

Parameters

node	the document node
name	the name for new node.

7.98.2.15 xmlNodePtr Xml::newTextChild (xmlNodePtr *node*, csr *name*, csr *text*) [static], [protected]

Create a new node with text content.

Parameters

node	the document node

name	the name for the node.
text	the content for the node.

7.98.2.16 void Xml::open (const Path & *fname*) [protected]

Open an existing XML file.

Parameters

fname	the name of the file to open.

7.98.2.17 void Xml::registerNamespace (const std::string & *prefix,* csr *uri*) [protected]

Call this to register the prefixes for all uri's used in XPath queries. The prefix does not have to be the same as used in the document and you must supply one for the "anonymous" uri's.

7.98.2.18 bool Xml::safeToSave ( )

Check if can save without clobbering other input.

Returns

true if its safe to save.

7.98.2.19 void Xml::save (const Path & *fname*)

Save the document to a new file.

Parameters

fname	the new file name.

7.98.2.20 void Xml::setCDATAContent (xmlNodePtr *node,* csr *text*) [protected]

Set the content for the node using CDATA.

Parameters

node	the document node
text	the value to assign to the node.

7.98.2.21 void Xml::setCompression (int *rate*)

Set the compression rate for saving the document.

Parameters

rate	the required compression factor.

7.98.2.22 void Xml::setContent (xmlNodePtr *node,* csr *text*) [protected]

Set the content for the node.

Parameters

node	the document node
text	the value to assign to the node content.

7.98.2.23 void SASSY::cfi::Xml::setDirty (bool *dt*) [inline]

Access to dirty flag if we change document outside of this interface.

Parameters

dt	new value for the dirty flag

7.98.2.24 void Xml::setDtd (csr *name*, const Path & *dtd*)

Set DTD identifier to dtd, which is just the dtd file name.

Parameters

name	the name for the DTD entry.
dtd	The DTD file name

7.98.2.25 void Xml::setProp (xmlNodePtr *node*, csr *prop*, csr *val*) [protected]

Set a string property.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.26 void Xml::setProp (xmlNodePtr *node*, csr *prop*, int *val*) [protected]

Set an integer property.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

7.98.2.27 void Xml::setProp (xmlNodePtr *node*, csr *prop*, double *val*) [protected]

Set a double property.

Parameters

node	the document node
prop	the property name for the element
val	the value for the property

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

- xml.h
- xml.cpp

Chapter 8

File Documentation

8.1 discover.h File Reference

Provides the API for objects that need to be discoverable for testing.

```
#include "stringy.h"
#include <map>
#include <vector>
#include <memory>
```

Classes

- struct SASSY::cfi::DiscoverPointer

  Holds a pointer to a discoverable object.
- class SASSY::cfi::Discoverable

  A mixin class that makes its owner discoverable.
- class SASSY::cfi::DiscoveryMgr

  Manager for discoverable objects.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

Macros

- #define **DISCOVERABLE**

Typedefs

- typedef std::shared_ptr
  < DiscoverPointer > SASSY::cfi::DiscoverSharedPointer

  A shared pointer that will be owned by a discoverable object.
- typedef std::weak_ptr
  < DiscoverPointer > SASSY::cfi::DiscoverWeakPointer

  A weak pointer that will be held by the DiscoveryMgr.

8.1.1 Detailed Description

Provides the API for objects that need to be discoverable for testing. This header should be included in the code for modules that need to be tested.

8.1.2 Macro Definition Documentation

8.1.2.1 #define DISCOVERABLE

Value:

```
SASSY::cfi::DiscoveryMgr::instance().save( __PRETTY_FUNCTION__, \
                        (discover.reset( new SASSY::cfi::DiscoverPointer( this )
      ), discover) );
```

8.2 except.h File Reference

An exception object with stream semantics.

```
#include <stdexcept>
#include <string>
#include <sstream>
#include <vector>
#include <errno.h>
#include <string.h>
```

Classes

- class rdf::vxt< N >

  An exception object with severity levels.

Namespaces

- rdf

  The namespace for the Resource Description Framework interface.

Macros

- #define VX(x) vx(x, __FILE__, __LINE__)

  Macro to include file name and line number in exception messages.

Typedefs

- using vx = rdf::vxt< rdf::rdf_class >

  A project specific exception class.

Enumerations

- enum rdf::Severity {
  rdf::Emergency, rdf::Alert, rdf::Critical, rdf::Error,
  rdf::Code, rdf::Warning, rdf::Notice, rdf::Info,
  rdf::Debug }

Severity levels for log messages.

8.2.1 Detailed Description

An exception object with stream semantics.

8.3 fdstream.h File Reference

Provide C++ stream interface to file descriptor integers.

```
#include <iostream>
#include <string>
```

Classes

- class SASSY::cfi::FD

  Wrap file descriptors in a class to ensure closed.
- class SASSY::cfi::fdoutbuf

  An output stream buffer.
- class SASSY::cfi::fdostream

  A file descriptor output stream.
- class SASSY::cfi::fdinbuf

  An input stream buffer.
- class SASSY::cfi::fdistream

  A file descriptor input stream.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.3.1 Detailed Description

Provide C++ stream interface to file descriptor integers.

8.4 ipc.h File Reference

Declarations for the semaphore component of libcfi.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include "stringy.h"
```

Classes

- class SASSY::cfi::Semaphore

  A semaphore for managing exclusive access to resources across multiple processes.
- class SASSY::cfi::SemaphoreLock

  Mutual exclusion lock.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.4.1 Detailed Description

Declarations for the semaphore component of libcfi.

8.5 log.h File Reference

Declarations for the logging component of libcfi.

```
#include <map>
#include <fstream>
#include "sx.h"
#include "udpsocket.h"
#include "ipc.h"
```

Classes

- class SASSY::cfi::logstream

  A stream class specialized for logging.
- class SASSY::cfi::logger

  An abstract base class used by the log stream to write logs.
- class SASSY::cfi::FormattingLogger

  Class for producing a formatted log message.
- class SASSY::cfi::StdLogger

  Class for logging to the std output streams.
- class SASSY::cfi::FileLogger

  Class for logging to a file.
- class SASSY::cfi::PlainFileLogger
- class SASSY::cfi::UDPLogger

  Class for logging to a logging server.
- class SASSY::cfi::TraceLogger

  Class for logging tracing output.
- class SASSY::cfi::SystemLogger

  logger that interfaces to the Linux syslog

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.5.1 Detailed Description

Declarations for the logging component of libcfi.

8.6 plugin.h File Reference

Declarations for the plug-in component of libcfi.

```
#include "stringy.h"
#include <map>
#include <list>
#include <memory>
#include <iostream>
```

Classes

- class SASSY::cfi::PlugIn

  Base class for objects loaded from dynamically loaded libraries.
- class SASSY::cfi::AutoRunPlugIn

  Base class for plug ins that are run immediately that the library is loaded.
- class SASSY::cfi::PlugInFactory

  Base class for factories that create plug-in objects.
- class SASSY::cfi::PlugInFamilyFactoryT< F >

  Template base class for families of plug-in factories.
- class SASSY::cfi::PlugInFactoryT< F, P >

  Template class for factories.
- struct SASSY::cfi::PlugInDescriptor

  Descriptor for plug-ins that can be used by the application.
- struct SASSY::cfi::PlugInDetails

  Details of plug-ins as provided by the loaded library.
- class SASSY::cfi::PlugInLib

  Manages an instance of a dynamically loaded shared library.
- class SASSY::cfi::PlugInMgr

  Manage the handling of plug-in shared libraries.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

Macros

- #define SASSY_PLUGIN_VERSION "0.1"

  Version number for plug-ins. This is checked when the library is loaded.

Typedefs

- typedef std::shared_ptr < PlugInLib > SASSY::cfi::PlugInLibPtr

  Shared pointer to PlugInLib.
- typedef std::unique_ptr < AutoRunPlugIn > SASSY::cfi::AutoRunPlugInPtr

  Unique pointer to an AutoRunPlugIn.

Functions

- void initSassyPlugin ()

  Initialise the plugin.
- void closeSassyPlugin ()

  Shutdown the plugin.

8.6.1 Detailed Description

Declarations for the plug-in component of libcfi. Applications will include this file if the need to load plug-ins such as used by the testing component.

Plug-in libraries will include this file.

8.6.2 Function Documentation

8.6.2.1 void closeSassyPlugin ( )

Shutdown the plugin.

Each plug-in will implement this function. It should delete the factory objects.

8.6.2.2 void initSassyPlugin ( )

Initialise the plugin.

Each plug-in will implement this function. It should add a PlugInDetails object to the PlugInMgr

8.7 proc.h File Reference

Declarations for classes that manage child processes.

```
#include <string>
#include <map>
#include <vector>
#include <signal.h>
#include <mutex>
```

Classes

- class SASSY::cfi::AbstractChildProcess

  Abstract child process interface for ChildProcessMgr.
- class SASSY::cfi::ChildProcess

  Represents the state of a child process.
- class SASSY::cfi::ProcessOwner

  Interface that allows the owner of a child process to be notified.
- class SASSY::cfi::ChildProcessMgr

  Manage the child processes.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.7.1 Detailed Description

Declarations for classes that manage child processes.

8.8 rdfxx.h File Reference

A C++ wrapper for Redland librdf.

```
#include <memory>
#include <string>
#include <map>
#include <rdfxx/except.h>
```

Classes

- class rdf::URI

  A shared pointer with constructors for the URI_ class.
- class rdf::Model

  A shared pointer with constructors for the Model_ class.
- class rdf::ResourceNode

  A shared pointer with constructors for the ResourceNode_ class.
- class rdf::LiteralNode

  A shared pointer with constructors for the LiteralNode_ class.
- class rdf::BlankNode

  A shared pointer with constructors for the BlankNode_ class.
- class rdf::Statement

  A shared pointer with constructors for the Statement_ class.
- class rdf::Parser

  A shared pointer with constructors for the Parser_ class.
- class rdf::Serializer

  A shared pointer with constructors for the Serializer_ class.

- class rdf::Stream

  A shared pointer with constructors for the Stream_ class.
- class rdf::Query

  A shared pointer with constructors for the Query_ class.
- class rdf::ErrorClient

  Client that is notified of errors and/or warnings.
- class rdf::Universe

  A singleton class responsible for managing the World objects.
- struct rdf::Format

  Instructions for converting a node to a string.
- class rdf::Prefixes

  Manages the prefixes and namespaces for a World.
- class rdf::Literal

  Hold the value, language and data type for an RDF literal.
- class rdf::World_

  An abstract class defining the methods for an RDF World.
- class rdf::Model_

  An abstract class defining the methods for an RDF Model.
- class rdf::Node_

  An abstract class defining the methods for an RDF Node.
- class rdf::ResourceNode_

  An abstract class defining the methods for an RDF Resource Node.
- class rdf::LiteralNode_

  An abstract class defining the methods for an RDF Literal Node.
- class rdf::BlankNode_

  An abstract class defining the methods for an RDF blank Node.
- class rdf::Parser_

  An abstract class defining the methods for an RDF Parser.
- class rdf::Query_

  An abstract class defining the methods for an RDF Query.
- class rdf::QueryResult_

  An abstract class defining the methods for an RDF Query Result.
- class rdf::QueryResults_

  An abstract class defining the methods for a set of Query Results.
- class rdf::QueryResults_::iterator

  Provide a C++ iterator over the results of a RDF query.
- class rdf::QueryString

  A class for assistingin the preparation of a SPARQL query.
- class rdf::Serializer_

  An abstract class defining the methods for an RDF Serializer.
- class rdf::Statement_

  An abstract class defining the methods for an RDF Statement.
- class rdf::Stream_

  An abstract class defining the methods for an RDF Stream.
- class rdf::URI_

  An abstract class defining the methods for an RDF URI.

Namespaces

- rdf

  The namespace for the Resource Description Framework interface.

Macros

- #define DEREF(A, b, c) deref< A##_, _##A, b >(c)

  Simplify the template calls for dereferencing a shared pointer.

Typedefs

- using rdf::World = std::shared_ptr< World_ >

  A shared pointer to a World object.
- using rdf::WorldRef = std::weak_ptr< World_ >

  A weak shared pointer to a World object.
- using rdf::NodeRef = std::weak_ptr< Node_ >

  A weak shared pointer to a Node object.
- using rdf::QueryResults = std::shared_ptr< QueryResults_ >

  A shared pointer to a set of query results.
- using rdf::StatementRef = std::weak_ptr< Statement_ >

  A weak shared pointer to a statement.
- using rdf::Node = std::shared_ptr< Node_ >

  A shared pointer to a Node object.

Enumerations

- enum rdf::Concept {
  Container, **Bag**, **Sequence**, **Alternative**,
  aboutEach, **List**, **first**, **rest**,
  nil, **Statement**, **object**, **predicate**,
  subject, **Resource**, **Class**, **subClassOf**,
  type, **Property**, **subPropertyOf**, **domain**,
  range, **ConstraintProperty**, **ConstraintResource**, **Description**,
  label, **seeAlso**, **comment**, **isDefinedBy** }

  An enumeration of RDF and RDFS concepts.
- enum rdf::DataType {
  UNDEF, **PlainLiteral**, **XMLLiteral**, **XHTML**,
  String, **Boolean**, **Decimal**, **Integer**,
  Double, **Float**, **Data**, **Time**,
  DateTime, **DateTimeStamp**, **Year**, **Month**,
  Day, **YearMonth**, **MonthDay**, **Duration**,
  YearMonthDuration, **DayTimeDuration**, **Byte**, **Short**,
  Int, **Long**, **UnsignedByte**, **UnsignedShort**,
  UnsignedLong, **PositiveInteger**, **NonNegativeInteger**, **NegativeInteger**,
  NonPositiveInteger, **HexBinary**, **Base64Binary**, **AnyURI**,
  Language, **NormalizedString**, **Token**, **NMTOKEN**,
  Name, **NCName** }

  A enumeration of the data types available for RDF literals.

Functions

- bool rdf::operator== (Statement, Statement)

  Check for equality of two statements.
- bool rdf::operator== (URI, URI)

  Check for equality of two URIs.
- template<class C , class P , typename T >
  T * rdf::deref (std::shared_ptr< C > a)

  A template function that converts a shared pointer into the corresponding librdf pointer.

8.8.1 Detailed Description

A C++ wrapper for Redland librdf. This file contains classes that provide a C++ interface to RDF models.

8.8.2 Macro Definition Documentation

8.8.2.1 #define DEREF(*A, b, c*) deref< A##_, _##A, b >(c)

Simplify the template calls for dereferencing a shared pointer.

Relies on the naming convention for classes described above. A is the shared pointer type, b is the librdf type and c is the shared pointer object.

8.9 stringy.h File Reference

Useful string functions.

```
#include <string>
#include <vector>
```

Classes

- class SASSY::Path

  Manipulate path strings.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.

Typedefs

- typedef const std::string & csr

  Save some time typing and shorten parameter lines.

Functions

- void SASSY::trim (std::string &s)

  rip off leading and trailing white spaces
- int SASSY::split (csr text, std::vector< std::string > &result)

  split a string into sub-strings at spaces
- std::string SASSY::expandMacros (csr s)

  expand a string containing $ macros

8.9.1 Detailed Description

Useful string functions.

8.10 sx.h File Reference

An exception object with stream semantics.

```
#include <stdexcept>
#include <string>
#include <sstream>
#include <vector>
#include <errno.h>
#include <string.h>
```

Classes

- class SASSY::sxt< N >

  An exception object with severity levels.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.

Macros

- #define SX(x) sx(x, __FILE__, __LINE__)

  Macro to include file name and line number in exception messages.

Typedefs

- using sx = SASSY::sxt< SASSY::sassy_class >

  A project specific exception class.

Enumerations

- enum SASSY::Severity {
  SASSY::Emergency, SASSY::Alert, SASSY::Critical, SASSY::Error,
  SASSY::Code, SASSY::Warning, SASSY::Notice, SASSY::Info,
  SASSY::Debug }

  Severity levels for log messages.

8.10.1 Detailed Description

An exception object with stream semantics.

8.11 test.h File Reference

Interface between applications and the test harness.

```
#include <string>
```

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.

Typedefs

- typedef void(∗ SASSY::AsyncTestEventFn)(const std::string &s)

  Pointer to function used to enqueue a test event.

Functions

- void SASSY::defaultAsyncTestEvent (const std::string &s)

  Function used to enqueue a test event.

Variables

- AsyncTestEventFn SASSY::asyncTestEvent = SASSY::defaultAsyncTestEvent

  Pointer to function used to enqueue a test event.

8.11.1 Detailed Description

Interface between applications and the test harness. This file should be included if the code is instrumented with event() calls that are used for asynchronous tests.

8.12 testmgr.h File Reference

Declarations for the test harness.

```
#include "stringy.h"
#include "sx.h"
#include "log.h"
#include "test.h"
#include <map>
#include <list>
#include <set>
#include <fstream>
#include <mutex>
#include <condition_variable>
#include <chrono>
```

Classes

- class SASSY::cdi::Tester

  Responsible for managing the testing.
- class SASSY::cdi::test_exception

  throw this to abandon a particular test case
- class SASSY::cdi::scenario_exception

  throw this to abandon an entire scenario
- class SASSY::cdi::TestEvent

Represent an event in the object under test.

- class SASSY::cdi::TestEventQueue

  Responsible for queuing TestEvents.
- class SASSY::cdi::TestFactory

  Responsible for creating an object that manages the resources for the entire test.
- class SASSY::cdi::TestFT< T >

  Responsible for creating a test object of the required type.
- class SASSY::cdi::AbstractTest

  Responsible for managing resources needed for the entire test.
- class SASSY::cdi::TestT< T >

  Responsible for installing a factory for making test objects.
- class SASSY::cdi::DefaultTest

  Provide a default version of the AbstractTest object.
- class SASSY::cdi::ScenarioFactory

  Define a type for scenario factories.
- class SASSY::cdi::ScenarioFT< T >

  Responsible for creating a scenario of some type.
- struct SASSY::cdi::ScenarioResults

  Responsible for storing the results of testing for a scenario.
- class SASSY::cdi::AbstractScenario

  Define a type for scenarios.
- class SASSY::cdi::ScenarioT< T >

  Responsible for installing a factory to create scenarios of the required type.
- class SASSY::cdi::DefaultScenario

  Provide a default scenario object.
- class SASSY::cdi::TestCaseFactory

  Define a base type for test case factories.
- class SASSY::cdi::TestCaseFT< T >

  Responsible for creating a test case of some type.
- class SASSY::cdi::AbstractTestCase

  Base class for test cases.
- class SASSY::cdi::TestCaseT< T >

  Responsible for installing a factory for the test case.
- class SASSY::cdi::AsyncTestCase

  Responsible for handling asynchronous tests.
- class SASSY::cdi::AsyncTestCaseT< T >

  Responsible for installing the factory for the test case type.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cdi

  The namespace for the Common Development Infrastructure.

8.12.1 Detailed Description

Declarations for the test harness.

8.13 trace.h File Reference

Support for tracing the execution path.

```
#include "stringy.h"
#include <map>
#include <sstream>
#include <mutex>
```

Classes

- class SASSY::cdi::CallTrace

  Class to create a call trace.
- class SASSY::cdi::NullTrace

  A class for dummy trace objects.
- class SASSY::cdi::Trace

  This class is used to create trace log entries.
- class SASSY::cdi::Tracer

  This class manages the tracing for an application.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cdi

  The namespace for the Common Development Infrastructure.

Macros

- #define TRACE(n) SASSY::cdi::NullTrace()

  Macro for adding trace statements to code.
- #define FN_TRACE(n, x)

  Macro for adding function call tracing to code.

8.13.1 Detailed Description

Support for tracing the execution path.

8.14 udpsocket.h File Reference

Declarations for a wrapper for the UDP socket.

```
#include "stringy.h"
#include <sys/socket.h>
#include <netinet/in.h>
```

Classes

- class SASSY::cfi::UDPSocket

  An abstract base class with common stuff for UDP sockets.
- class SASSY::cfi::UDPClientSocket

  A UDP client socket.
- class SASSY::cfi::UDPServerSocket

  A UDP Server socket.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.14.1 Detailed Description

Declarations for a wrapper for the UDP socket.

8.15 xini.h File Reference

A class for XML based configuration file.

```
#include "xml.h"
```

Classes

- class SASSY::cfi::XINI

  Load an XML configuration file.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

8.15.1 Detailed Description

A class for XML based configuration file.

8.16 xml.h File Reference

A simple C++ wrapper for libxml2.

```
#include <libxml/tree.h>
#include <libxml/xpath.h>
#include <sys/types.h>
#include <sys/stat.h>
#include "stringy.h"
#include <vector>
```

Classes

- class SASSY::cfi::Xml

  A C++ wrapper for libxml2.

Namespaces

- SASSY

  The namespace for the Software Architecture Support System project.
- SASSY::cfi

  The namespace for the Common Facilities Infrastructure components.

Macros

- #define SASSY_RELEASE "V0.1"

  Macro specifying the release version of Sassy.

8.16.1 Detailed Description

A simple C++ wrapper for libxml2.

Index