Hosting Environment (Daemon) Services

Generated by Doxygen 1.5.7.1

Tue Apr 28 11:10:42 2009

Contents

Chapter 1

Data Structure Index

1.1 Class Hierarchy

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

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 Arc::AREXClient Class Reference

A client class for the A-REX service.

`#include <arex_client.h>`

Public Member Functions

- **AREXClient** (std::string configFile="") throw (AREXClientError)
- ~**AREXClient** ()
- std::string **submit** (std::istream &jsdl_file, AREXFileList &file_list, bool delegate=false) throw (AREXClientError)
- std::string **stat** (const std::string &jobid) throw (AREXClientError)
- void **kill** (const std::string &jobid) throw (AREXClientError)
- void **clean** (const std::string &jobid) throw (AREXClientError)
- std::string **sstat** (void) throw (AREXClientError)

4.1.1 Detailed Description

A client class for the A-REX service.

This class is a client for the A-REX service (Arc Resource-coupled EXecution service). It provides methods for three operations on an A-REX service:

- **Job** (p. 22) submission
- **Job** (p. 22) status queries
- **Job** (p. 22) termination

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Arc::AREXClient::AREXClient (std::string *configFile* = "") throw (AREXClientError)

The constructor for the **AREXClient** (p. 9) class.

This is the constructor for the **AREXClient** (p. 9) class. It creates an A-REX client that corresponds to a specific A-REX service, which is specified in a configuration file. The configuration file also specifies how to set up the communication chain for the client. The location of the configuration file can be provided as a parameter to the method. If no such parameter is given, the environment variable ARC_AREX_CONFIG is assumed to contain the location. If there is no such environment variable, the configuration file is assumed to be "arex_client.xml" in the current working directory.

Parameters:

configFile The location of the configuration file.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

4.1.2.2 Arc::AREXClient::~AREXClient ()

The destructor.

This is the destructor. It does what destructors usually do, cleans up...

4.1.3 Member Function Documentation

4.1.3.1 void Arc::AREXClient::clean (const std::string & *jobid*) throw (AREXClientError)

Removes a job.

This method sends a request to the A-REX service to remove a job from it's pool. If job is running it will be killed by service as well.

Parameters:

jobid The **Job** (p. 22) ID of the job to remove.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

4.1.3.2 void Arc::AREXClient::kill (const std::string & *jobid*) throw (AREXClientError)

Terminates a job.

This method sends a request to the A-REX service to terminate a job.

Parameters:

jobid The **Job** (p. 22) ID of the job to terminate.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

4.1.3.3 std::string Arc::AREXClient::sstat (void) throw (AREXClientError)

Query the status of a service.

This method queries the A-REX service about it's status.

Returns:

The XML document representing status of the service.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

4.1.3.4 std::string Arc::AREXClient::stat (const std::string & *jobid*) throw (AREXClientError)

Query the status of a job.

This method queries the A-REX service about the status of a job.

Parameters:

jobid The **Job** (p. 22) ID of the job.

Returns:

The status of the job.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

4.1.3.5 std::string Arc::AREXClient::submit (std::istream & *jsdl_file*, AREXFileList & *file_list*, bool *delegate* = `false`) throw (AREXClientError)

Submit a job.

This method submits a job to the A-REX service corresponding to this client instance.

Parameters:

jsdl_file An input stream from which the JSDL file for the job can be read.

Returns:

The **Job** (p. 22) ID of the the submitted job.

Exceptions:

An **AREXClientError** (p. 12) object if an error occurs.

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

- arex_client.h

4.2 Arc::AREXClientError Class Reference

An exception class for the **AREXClient** (p. 9) class.

`#include <arex_client.h>`

Public Member Functions

- **AREXClientError** (const std::string &what="")

4.2.1 Detailed Description

An exception class for the **AREXClient** (p. 9) class.

This is an exception class that is used to handle runtime errors discovered in the **AREXClient** (p. 9) class.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Arc::AREXClientError::AREXClientError (const std::string & *what* = "")

Constructor.

This is the constructor of the **AREXClientError** (p. 12) class.

Parameters:

what An explanation of the error.

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

- arex_client.h

4.3 ARex::ARexJob Class Reference

`#include <job.h>`

Public Member Functions

- **ARexJob** (const std::string &id, ARexGMConfig &config, Arc::Logger &logger, bool fast_auth_check=false)
- **ARexJob** (Arc::XMLNode jsdl, ARexGMConfig &config, const std::string &credentials, const std::string &clientid, Arc::Logger &logger, const Arc::XMLNode &migration=Arc::XMLNode())
- std::string **Failure** (void)
- std::string **ID** (void)
- bool **GetDescription** (Arc::XMLNode &jsdl)
- bool **Cancel** (void)
- bool **Clean** (void)
- bool **Resume** (void)
- std::string **State** (void)
- std::string **State** (bool &job_pending)
- bool **Failed** (void)
- std::string **SessionDir** (void)
- int **CreateFile** (const std::string &filename)
- bool **UpdateCredentials** (const std::string &credentials)

Static Public Member Functions

- static int **TotalJobs** (ARexGMConfig &config, Arc::Logger &logger)
- static std::list< std::string > **Jobs** (ARexGMConfig &config, Arc::Logger &logger)

4.3.1 Detailed Description

This class represents convenience interface to manage jobs handled by Grid Manager. It works mostly through corresponding classes and functions of Grid Manager.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ARex::ARexJob::ARexJob (const std::string & *id*, ARexGMConfig & *config*, Arc::Logger & *logger*, bool *fast_auth_check* = `false`)

Create instance which is an interface to existing job

4.3.2.2 ARex::ARexJob::ARexJob (Arc::XMLNode *jsdl*, ARexGMConfig & *config*, const std::string & *credentials*, const std::string & *clientid*, Arc::Logger & *logger*, const Arc::XMLNode & *migration* = `Arc::XMLNode()`)

Create new job with provided JSDL description

4.3.3 Member Function Documentation

4.3.3.1 bool ARex::ARexJob::Cancel (void)

Cancel processing/execution of job

4.3.3.2 bool ARex::ARexJob::Clean (void)

Remove job from local pool

4.3.3.3 int ARex::ARexJob::CreateFile (const std::string & *filename*)

Creates file in job's session directory and returns handler

4.3.3.4 bool ARex::ARexJob::Failed (void)

Returns true if job has failed

4.3.3.5 std::string ARex::ARexJob::Failure (void) [inline]

Returns textual description of failure of last operation

4.3.3.6 bool ARex::ARexJob::GetDescription (Arc::XMLNode & *jsdl*)

Fills provided jsdl with job description

4.3.3.7 std::string ARex::ARexJob::ID (void) [inline]

Return ID assigned to job

4.3.3.8 static std::list<std::string> ARex::ARexJob::Jobs (ARexGMConfig & *config*, Arc::Logger & *logger*) [static]

Returns list of user's jobs. Fine-grained ACL is ignored.

4.3.3.9 bool ARex::ARexJob::Resume (void)

Resume execution of job after error

4.3.3.10 std::string ARex::ARexJob::SessionDir (void)

Returns path to session directory

4.3.3.11 std::string ARex::ARexJob::State (bool & *job_pending*)

Returns current state of job and sets job_pending to true if job is pending due to external limits

4.3.3.12 std::string ARex::ARexJob::State (void)

Returns current state of job

4.3.3.13 static int ARex::ARexJob::TotalJobs (ARexGMConfig & *config*, Arc::Logger & *logger*) [static]

Return number of jobs associated with this configuration. TODO: total for all user configurations.

4.3.3.14 bool ARex::ARexJob::UpdateCredentials (const std::string & *credentials*)

Updates job credentials

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

- a-rex/job.h

4.4 CacheConfig Class Reference

`#include <conf_cache.h>`

Public Member Functions

- **CacheConfig** (std::string username="")
- void **setCacheDirs** (std::list< std::string > cache_dirs)

4.4.1 Detailed Description

Reads conf file and provides methods to obtain cache info from it.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CacheConfig::CacheConfig (std::string *username* = "")

Create a new **CacheConfig** (p. 16) instance. Read the config file and fill in private member variables with cache parameters. If different users are defined in the conf file, use the cache parameters for the given username.

4.4.3 Member Function Documentation

4.4.3.1 void CacheConfig::setCacheDirs (std::list< std::string > *cache_dirs*) [inline]

To allow for substitutions done during configuration

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

- conf_cache.h

4.5 CacheConfigException Class Reference

`#include <conf_cache.h>`

4.5.1 Detailed Description

Exception thrown by constructor caused by bad cache params in conf file

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

- conf_cache.h

4.6 ArcSec::Charon Class Reference

`#include <charon.h>`

4.6.1 Detailed Description

A Service which includes the ArcPDP functionality; it can be deployed as an independent service to provide request evaluation functionality for the other remote services

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

- charon.h

4.7 ARex::Config Class Reference

`#include <configcore.h>`

Public Member Functions

- const std::list< ConfGrp > & **GetConfigs** () const
- std::list< std::string > **ConfValue** (const std::string &path) const
- std::string **FirstConfValue** (const std::string &path) const

4.7.1 Detailed Description

Core configuration class.

4.7.2 Member Function Documentation

4.7.2.1 std::list<std::string> ARex::Config::ConfValue (const std::string & *path*) const

Get the configuration values from key.

4.7.2.2 std::string ARex::Config::FirstConfValue (const std::string & *path*) const

Get the first configuration value from key. This is meant as a short cut when it is known that the key is not multivalued.

4.7.2.3 const std::list<ConfGrp>& ARex::Config::GetConfigs () const

Returns the parsed options.

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

- **configcore.h**

4.8 ARex::ConfigError Class Reference

`#include <configcore.h>`

Public Member Functions

- **ConfigError** (std::string message)

4.8.1 Detailed Description

Error configuration class.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ARex::ConfigError::ConfigError (std::string *message*) [inline]

Constructor for the **ConfigError** (p. 20) exception. Calls the corresponding constructor in ARCLibError.

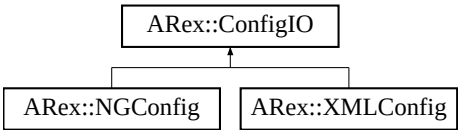
The documentation for this class was generated from the following file:

- **configcore.h**

4.9 ARex::ConfigIO Class Reference

`#include <configio.h>`

Inheritance diagram for ARex::ConfigIO::



Public Member Functions

- virtual **Config Read** (std::istream &is)=0
- virtual void **Write** (const **Config** &conf, std::ostream &os)=0

4.9.1 Detailed Description

Virtual base-class for reading and writing configuration files. Concrete instances include **NGConfig** (p. 37) and **XMLConfig** (p. 51).

4.9.2 Member Function Documentation

4.9.2.1 virtual Config ARex::ConfigIO::Read (std::istream & *is*) [pure virtual]

Read the named configuration source.

Implemented in **ARex::NGConfig** (p. 37), and **ARex::XMLConfig** (p. 51).

4.9.2.2 virtual void ARex::ConfigIO::Write (const Config & *conf*, std::ostream & *os*) [pure virtual]

Write configuration to named configuration destination.

Implemented in **ARex::NGConfig** (p. 37), and **ARex::XMLConfig** (p. 51).

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

- configio.h

4.10 Job Class Reference

`#include <job.h>`

Public Member Functions

- **Job** (void)
- **Job** (std::string path)
- ~**Job** (void)
- **operator bool** (void)
- bool **Start** (void)
- bool **Cancel** (void)
- bool **Resume** (void)
- std::string **GetState** (void)
- std::string **GetSessionDir** (void)

4.10.1 Detailed Description

Collect all information (status, lrms info, user) required to handle job

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Job::Job (void)

Constructor: Creates empty job

4.10.2.2 Job::Job (std::string *path*)

Constructor: load job information form files

4.10.2.3 Job::~Job (void)

Destuctior

4.10.3 Member Function Documentation

4.10.3.1 bool Job::Cancel (void)

Cancel processing/execution of job

4.10.3.2 std::string Job::GetSessionDir (void)

Returns the session directory of the job

4.10.3.3 std::string Job::GetState (void)

Returns the string represnetation of job state

4.10.3.4 Job::operator bool (void) [inline]

Helper logical operators

4.10.3.5 bool Job::Resume (void)

Resume execution of job after error

4.10.3.6 bool Job::Start (void)

Starts job. Most of the cases it means to submit to LRMS

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

- arex2/job.h

4.11 ARex2::JobControl Class Reference

`#include <job_control.h>`

4.11.1 Detailed Description

Represents job controll information like session dir, control dir

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

- job_control.h

4.12 ARex2::JobDataCache Class Reference

`#include <job_data_cache.h>`

4.12.1 Detailed Description

Data cache

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

- job_data_cache.h

4.13 ARex2::JobDescription Class Reference

`#include <job_descr.h>`

Data Structures

- class **InputFile**
- class **Notification**
- class **OutputFile**

Public Member Functions

- std::string & **JobName** (void)

4.13.1 Detailed Description

Internal representation of **Job** (p. 22) described by JSDL

4.13.2 Member Function Documentation

4.13.2.1 std::string& ARex2::JobDescription::JobName (void) [inline]

Interface methods to access stored values.

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

- job_descr.h

4.14 ARex2::JobDescription::InputFile Class Reference

`#include <job_descr.h>`

4.14.1 Detailed Description

Class represents the one of the input file of the job

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

- job_descr.h

4.15 ARex2::JobDescription::Notification Class Reference

`#include <job_descr.h>`

4.15.1 Detailed Description

Class represents notification requiest

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

- job_descr.h

4.16 ARex2::JobDescription::OutputFile Class Reference

`#include <job_descr.h>`

4.16.1 Detailed Description

Class represents the one of the output file of the job

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

- job_descr.h

4.17 JobLog Class Reference

`#include <job_log.h>`

4.17.1 Detailed Description

Put short information into log when every job starts/finishes. And store more detailed information for Reporter.

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

- job_log.h

4.18 ARex2::JobLRMSInfo Class Reference

`#include <lrms.h>`

4.18.1 Detailed Description

Class represents the information about job in LRMS

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

- lrms.h

4.19 JobRequestError Class Reference

`#include <job.h>`

Public Member Functions

- **JobRequestError** (std::string message)

4.19.1 Detailed Description

Exception class thrown in case of errors with the JobRequest class.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 JobRequestError::JobRequestError (std::string *message*) [inline]

Standard exception class constructor.

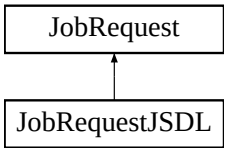
The documentation for this class was generated from the following file:

- a-rex/grid-manager/jobdesc/job.h

4.20 JobRequestJSDL Class Reference

`#include <job_jsdl.h>`

Inheritance diagram for JobRequestJSDL::



4.20.1 Detailed Description

Class to represent the request for computational job.

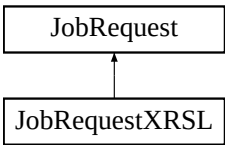
The documentation for this class was generated from the following file:

- job_jsdl.h

4.21 JobRequestXRSL Class Reference

`#include <job_xrsl.h>`

Inheritance diagram for JobRequestXRSL::



4.21.1 Detailed Description

Class to represent the request for computational job.

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

- job_xrsl.h

4.22 ARex2::JobState Class Reference

`#include <job_state.h>`

4.22.1 Detailed Description

Represents the state of job. It includes error messages as well

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

- job_state.h

4.23 ARex2::JobUser Class Reference

`#include <job_user.h>`

4.23.1 Detailed Description

Job (p. 22) run under the privilages of one of the system user. This class collects information related to this user

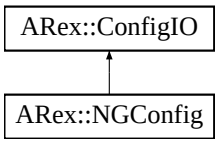
The documentation for this class was generated from the following file:

- job_user.h

4.24 ARex::NGConfig Class Reference

`#include <ngconfig.h>`

Inheritance diagram for ARex::NGConfig::



Public Member Functions

- **Config Read** (std::istream &is)
- void **Write** (const **Config** &config, std::ostream &os)

4.24.1 Detailed Description

Configuration class used for reading configuration files ARC-style.

4.24.2 Member Function Documentation

4.24.2.1 Config ARex::NGConfig::Read (std::istream & *is*) [virtual]

Read old arc.conf style configuration.

Implements **ARex::ConfigIO** (p. 21).

4.24.2.2 void ARex::NGConfig::Write (const Config & *config*, std::ostream & *os*) [virtual]

Write configuration to named file.

Implements **ARex::ConfigIO** (p. 21).

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

- ngconfig.h

4.25 Hopi::PayloadFile Class Reference

`#include <PayloadFile.h>`

Public Member Functions

- **PayloadFile** (const char *filename)
- **PayloadFile** (const char *filename, int size)

4.25.1 Detailed Description

Implementation of PayloadRawInterface which provides access to ordinary file. Currently only read-only mode is supported.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Hopi::PayloadFile::PayloadFile (const char * *filename*)

Creates object associated with file for reading from it

4.25.2.2 Hopi::PayloadFile::PayloadFile (const char * *filename*, int *size*)

Creates object associated with file for writing into it. Use size=-1 for undefined size.

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

- hopi/PayloadFile.h

4.26 ARex::PayloadFile Class Reference

`#include <PayloadFile.h>`

Public Member Functions

- **PayloadFile** (const char *filename, size_t start=0, size_t end=(size_t)(-1))
- **PayloadFile** (int handle, size_t start=0, size_t end=(size_t)(-1))
- **PayloadFile** (const char *filename, int size)

4.26.1 Detailed Description

Implementation of PayloadRawInterface which provides access to ordinary file. Currently only read-only mode is supported.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 ARex::PayloadFile::PayloadFile (const char * *filename*, size_t *start* = `0`, size_t *end* = `(size_t)(-1)`)

Creates object associated with file for reading from it

4.26.2.2 ARex::PayloadFile::PayloadFile (int *handle*, size_t *start* = `0`, size_t *end* = `(size_t)(-1)`)

Creates object associated with file handle for reading or writing to it

4.26.2.3 ARex::PayloadFile::PayloadFile (const char * *filename*, int *size*)

Creates object associated with file for writing into it. Use size=-1 for undefined size.

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

- a-rex/PayloadFile.h

4.27 RTE Class Reference

`#include <rte.h>`

Public Member Functions

- **RTE** (const std::string &re)
- ~**RTE** ()
- std::string **str** () const
- std::string **Name** () const
- std::string **Version** () const
- bool **operator==** (const **RTE** &other) const
- bool **operator!=** (const **RTE** &other) const
- bool **operator**> (const **RTE** &other) const
- bool **operator**< (const **RTE** &other) const
- bool **operator**>**=** (const **RTE** &other) const
- bool **operator**<**=** (const **RTE** &other) const

4.27.1 Detailed Description

Interface for handling runtime environments. **RTE** (p. 40) class. It represents a runtime environment, and provides functionality for getting information about them.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 RTE::RTE (const std::string & *re*)

Constructs a new runtime environemt. String should in general be of the type: STRING-VERSION. Where version consists of numbers with . between them.

4.27.2.2 RTE::~RTE ()

Destructor. Not that much to say.

4.27.3 Member Function Documentation

4.27.3.1 std::string RTE::Name () const

Returns the name of the runtime environment.

4.27.3.2 bool RTE::operator!= (const RTE & *other*) const

Inequility operator. Return the opsite of ==

4.27.3.3 bool RTE::operator< (const RTE & *other*) const

Less than operator. Returns false if the other is equal, otherwise it returns the opposite of >

4.27.3.4 bool RTE::operator<= (const RTE & *other*) const

Less than or equal operator. Returns the oppsite of >

4.27.3.5 bool RTE::operator== (const RTE & *other*) const

Equliaty operator. Returns true if the runtime environments have the string representation.

4.27.3.6 bool RTE::operator> (const RTE & *other*) const

Greater than operator. Returns true if the compared runtime environment is greater than the current.

4.27.3.7 bool RTE::operator>= (const RTE & *other*) const

Greater or equal operator. Returns the opposite of <

4.27.3.8 std::string RTE::str () const

Returns a string representation of the runtime environment. This is usually the same as given in the constructor.

4.27.3.9 std::string RTE::Version () const

Returns the version of the runtime environment.

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

- rte.h

4.28 ArcSec::Service_AA Class Reference

`#include <aaservice.h>`

4.28.1 Detailed Description

A Service which includes the AttributeAuthority functionality; it accepts the <samlp:AttributeQuery> which includes the <Subject> of the principal from the request and <Attribute> which the request would get; it access some local attribute database and returns <samlp:Assertion> which includes the <Attribute>

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

- aaservice.h

4.29 Compiler::Service_Compiler Class Reference

`#include <compiler.h>`

Public Member Functions

- **Service_Compiler** (Arc::Config *cfg)
- virtual Arc::MCC_Status **process** (Arc::Message &, Arc::Message &)

4.29.1 Detailed Description

This service need in the server config a "compiler:scriptfile_url" element. It is the scriptfile place, where-from it will be download the JSDL.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 Compiler::Service_Compiler::Service_Compiler (Arc::Config * *cfg*)

Constructor accepts configuration describing content of scriptfile_url

4.29.3 Member Function Documentation

4.29.3.1 virtual Arc::MCC_Status Compiler::Service_Compiler::process (Arc::Message &, Arc::Message &) [virtual]

Service request processing routine

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

- compiler.h

4.30 ArcSec::Service_Delegation Class Reference

A Service which launches the proxy certificate request; it accepts the request from.

`#include <delegation.h>`

4.30.1 Detailed Description

A Service which launches the proxy certificate request; it accepts the request from.

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

- delegation.h

4.31 ArcSec::Service_SLCS Class Reference

`#include <slcs.h>`

4.31.1 Detailed Description

A Service which signs the short-lived certificate; it accepts the certificate request from from client side through soap, signs a short-lived certificate and sends back through soap. This service is supposed to be deployed together with the SPService and saml2sso.serviceprovider handler, in order to sign certificate based on the authentication result from saml2sso profile. Also the saml attribute (inside the saml assertion from saml2sso profile) will be put into the signed short-lived certificate. By deploying this service together with SPService and saml2sso.serviceprovider handler, we can get the convertion from username/password ——> x509 certificate.

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

- slcs.h

4.32 SPService::Service_SP Class Reference

`#include <SPService.h>`

Public Member Functions

- **Service_SP** (Arc::Config *cfg)
- virtual Arc::MCC_Status **process** (Arc::Message &, Arc::Message &)

4.32.1 Detailed Description

This is service which accepts HTTP request from user agent (web browser) in the client side and processes the functionality of Service Provider in SAML2 SSO profile — composing <AuthnRequest> Note: the IdP name is provided by the user agent directely when it gives a request, instead of the WRYF(where are you from) or Discovery Service in other implementation

4.32.2 Constructor & Destructor Documentation

4.32.2.1 SPService::Service_SP::Service_SP (Arc::Config * *cfg*)

Constructor

4.32.3 Member Function Documentation

4.32.3.1 virtual Arc::MCC_Status SPService::Service_SP::process (Arc::Message &, Arc::Message &) [virtual]

Service request processing routine

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

- SPService.h

4.33 Time Class Reference

`#include <datetime.h>`

Public Member Functions

- **Time** ()
- **Time** (const time_t &)
- **Time** (const std::string &)
- **Time** & **operator=** (const time_t &)
- void **SetTime** (const time_t &)
- time_t **GetTime** () const
- **operator std::string** () const
- std::string **str** (const TimeFormat &=time_format) const
- bool **operator**< (const **Time** &) const
- bool **operator**> (const **Time** &) const
- bool **operator**<= (const **Time** &) const
- bool **operator**>= (const **Time** &) const
- bool **operator==** (const **Time** &) const
- bool **operator!=** (const **Time** &) const

Static Public Member Functions

- static void **SetFormat** (const TimeFormat &)
- static TimeFormat **GetFormat** ()

4.33.1 Detailed Description

A class for storing and manipulating times.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 Time::Time ()

Default constructor. The time is put equal the current time.

4.33.2.2 Time::Time (const time_t &)

Constructor that takes a time_t variable and stores it.

4.33.2.3 Time::Time (const std::string &)

Constructor that tries to convert a string into a time_t.

4.33.3 Member Function Documentation

4.33.3.1 static TimeFormat Time::GetFormat () [static]

Gets the default format for time strings.

4.33.3.2 time_t Time::GetTime () const

gets the time

4.33.3.3 Time::operator std::string () const

Returns a string representation of the time, using the default format.

4.33.3.4 bool Time::operator!= (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.5 bool Time::operator< (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.6 bool Time::operator<= (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.7 Time& Time::operator= (const time_t &)

Assignment operator from a time_t.

4.33.3.8 bool Time::operator== (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.9 bool Time::operator> (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.10 bool Time::operator>= (const Time &) const

Comparing two **Time** (p. 47) objects.

4.33.3.11 static void Time::SetFormat (const TimeFormat &) [static]

Sets the default format for time strings.

4.33.3.12 void Time::SetTime (const time_t &)

sets the time

4.33.3.13 std::string Time::str (const TimeFormat & = `time_format`) const

Returns a string representation of the time, using the specified format.

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

- datetime.h

4.34 TimeError Class Reference

`#include <datetime.h>`

Public Member Functions

- **TimeError** (std::string message)

4.34.1 Detailed Description

Class to represent errors thrown by the **Time** (p. 47) class.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 TimeError::TimeError (std::string *message*) [inline]

Standard exception class constructor.

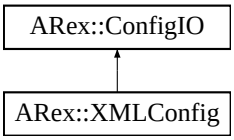
The documentation for this class was generated from the following file:

- datetime.h

4.35 ARex::XMLConfig Class Reference

`#include <xmlconfig.h>`

Inheritance diagram for ARex::XMLConfig::



Public Member Functions

- **Config Read** (std::istream &is)
- void **Write** (const **Config** &config, std::ostream &os)

4.35.1 Detailed Description

Class for reading in configuration files in xml-format. It uses libxml2 for xml-parsing.

4.35.2 Member Function Documentation

4.35.2.1 Config ARex::XMLConfig::Read (std::istream & *is*) [virtual]

Read configuration.

Implements **ARex::ConfigIO** (p. 21).

4.35.2.2 void ARex::XMLConfig::Write (const Config & *config*, std::ostream & *os*) [virtual]

Write configuration.

Implements **ARex::ConfigIO** (p. 21).

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

- xmlconfig.h

4.36 Xrsl Class Reference

`#include <xrsl.h>`

Public Member Functions

- **Xrsl** (const std::string &xrsl_string) throw (XrslError)
- **Xrsl** (xrsl_operator=operator_and)
- **Xrsl** (globus_rsl_t *)
- **Xrsl** (const **Xrsl** &other_xrsl)
- **Xrsl** & **operator=** (const **Xrsl** &other_xrsl)
- ~**Xrsl** ()
- void **Print** () const
- const std::string **str** () const throw (XrslError)
- std::list< **Xrsl** > **SplitMulti** ()
- std::list< **Xrsl** > **SplitOrRelation** () throw (XrslError)
- void **AddRelation** (const **XrslRelation** &relation, bool force=true) throw (XrslError)
- void **AddSimpleRelation** (const std::string &attr, xrsl_operator op, const std::string &val, bool force=true) throw (XrslError)
- void **AddXrsl** (**Xrsl** &axrsl) throw (XrslError)
- **XrslRelation GetRelation** (const std::string &attr) throw (XrslError)
- std::list< **XrslRelation** > **GetAllRelations** (const std::string &attr)
- bool **IsRelation** (const std::string &)
- void **RemoveRelation** (const std::string &attr) throw (XrslError)
- void **Validate** (const std::list< **XrslValidationData** > &valid_attributes, bool allow_unknown=false) throw (XrslError)
- void **Eval** ()

4.36.1 Detailed Description

Class used to simplify manipulation of xRSL job descriptions.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 Xrsl::Xrsl (const std::string & *xrsl_string*) throw (XrslError)

Constructs a **Xrsl** (p. 52) object from a string respresentation.

4.36.2.2 Xrsl::Xrsl (xrsl_operator = `operator_and`)

Constructs empty **Xrsl** (p. 52) object.

4.36.2.3 Xrsl::Xrsl (globus_rsl_t *)

Construct **Xrsl** (p. 52) object from globus_rsl_t* .

4.36.2.4 Xrsl::Xrsl (const Xrsl & *other_xrsl*)

Copy constructor.

4.36.2.5 Xrsl::~Xrsl ()

Destructor.

4.36.3 Member Function Documentation

4.36.3.1 void Xrsl::AddRelation (const XrslRelation & *relation*, bool *force* = `true`) throw (XrslError)

Adds a new relation. Throws exception if relation already exists in the xrsl and force is not true.

4.36.3.2 void Xrsl::AddSimpleRelation (const std::string & *attr*, xrsl_operator *op*, const std::string & *val*, bool *force* = `true`) throw (XrslError)

Adds simple relation specified by attribute, xrsl-operator and value. Throws exception if relation already exists in the xrsl and force is not true.

4.36.3.3 void Xrsl::AddXrsl (Xrsl & *axrsl*) throw (XrslError)

Adds a sub-Xrsl to the **Xrsl** (p. 52).

4.36.3.4 void Xrsl::Eval ()

Performs RSL alias substitution etc.

4.36.3.5 std::list<XrslRelation> Xrsl::GetAllRelations (const std::string & *attr*)

Get all XrslRelation's in the xrsl with attribute equal to parameter attr.

4.36.3.6 XrslRelation Xrsl::GetRelation (const std::string & *attr*) throw (XrslError)

Gets the first **XrslRelation** (p. 56) corresponding to the attribute.

4.36.3.7 bool Xrsl::IsRelation (const std::string &)

Does the relation with this attribute exist?

4.36.3.8 Xrsl& Xrsl::operator= (const Xrsl & *other_xrsl*)

Copy-assignment constructor.

4.36.3.9 void Xrsl::Print () const

Print detailed information about each relation.

4.36.3.10 void Xrsl::RemoveRelation (const std::string & *attr*) throw (XrslError)

Removes a relation. Throws an exception if the relation does not exist in the xrsl. The relation may be of any type.

4.36.3.11 std::list<Xrsl> Xrsl::SplitMulti ()

If the **Xrsl** (p. 52) start with a +, split the **Xrsl** (p. 52) into multiple Xrsls.

4.36.3.12 std::list<Xrsl> Xrsl::SplitOrRelation () throw (XrslError)

Splits an **Xrsl** (p. 52) containing or-operators into separate Xrsl's. Example: &(executable=/bin/echo)(|(cluster=cl1)(cluster=cl2)) split into &(executable=/bin/echo)(cluster=cl1) and &(executable=/bin/echo)(cluster=cl2).

4.36.3.13 const std::string Xrsl::str () const throw (XrslError)

Converts the **Xrsl** (p. 52) object to std::string representation.

4.36.3.14 void Xrsl::Validate (const std::list< XrslValidationData > & *valid_attributes*, bool *allow_unknown* = `false`) throw (XrslError)

Ensures that the xrsl only contains valid attributes. Throws exception if some attribute has invalid format, or that some mandatory attribute is missing.

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

- xrsl.h

4.37 XrslError Class Reference

`#include <xrsl.h>`

Public Member Functions

- **XrslError** (std::string message_arg)

4.37.1 Detailed Description

Class represents exceptions associated with usage of the **Xrsl** (p. 52) class.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 XrslError::XrslError (std::string *message_arg*) [inline]

Standard exception constructor.

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

- xrsl.h

4.38 XrslRelation Class Reference

`#include <xrsl.h>`

Public Member Functions

- **XrslRelation** (const std::string &attribute, const xrsl_operator&oper, const std::string &value)
- **XrslRelation** (const std::string &attribute, const xrsl_operator&oper, const std::list< std::string > &value)
- **XrslRelation** (const std::string &attribute, const xrsl_operator&oper, const std::list< std::list< std::string > > &value)
- **XrslRelation** (globus_rsl_t *relation)
- **XrslRelation** (const **XrslRelation** &other_relation)
- ~**XrslRelation** ()
- std::string **str** ()
- std::string **GetAttribute** () const
- xrsl_operator **GetOperator** () const
- std::string **GetSingleValue** () throw (XrslError)
- std::list< std::string > **GetListValue** () throw (XrslError)
- std::list< std::list< std::string > > **GetDoubleListValue** () throw (XrslError)
- globus_rsl_t * **GetRelation** () const

4.38.1 Detailed Description

XrslRelation (p. 56) class that describes an Xrsl-relation with an attribute, an operator and a value. Various constructors and methods for extracting the attribute and the value (single value or list) are given.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 XrslRelation::XrslRelation (const std::string & *attribute*, const xrsl_operator& *oper*, const std::string & *value*)

Constructor constructing an xrsl-relation from an attribute, an operator and a value.

4.38.2.2 XrslRelation::XrslRelation (const std::string & *attribute*, const xrsl_operator& *oper*, const std::list< std::string > & *value*)

Constructor constructing an xrsl-relation from an attribute, an operator and a value-list.

4.38.2.3 XrslRelation::XrslRelation (const std::string & *attribute*, const xrsl_operator& *oper*, const std::list< std::list< std::string > > & *value*)

Constructor constructing an xrsl-relation from an attribute, an operator and a double value-list.

4.38.2.4 XrslRelation::XrslRelation (globus_rsl_t * *relation*)

Constructs a relation from a globus_rsl_t*.

4.38.2.5 XrslRelation::XrslRelation (const XrslRelation & *other_relation*)

Copy-constructor.

4.38.2.6 XrslRelation::~XrslRelation ()

Destructor.

4.38.3 Member Function Documentation

4.38.3.1 std::string XrslRelation::GetAttribute () const

Returns the attribute of the relation.

4.38.3.2 std::list<std::list<std::string> > XrslRelation::GetDoubleListValue () throw (XrslError)

If the value of the relation is a double list value, return it.

4.38.3.3 std::list<std::string> XrslRelation::GetListValue () throw (XrslError)

If the value of the relation is a list value, return it.

4.38.3.4 xrsl_operator XrslRelation::GetOperator () const

Returns the xrsl_operator of the attribute.

4.38.3.5 globus_rsl_t* XrslRelation::GetRelation () const

Returns relation.

4.38.3.6 std::string XrslRelation::GetSingleValue () throw (XrslError)

If the value of the representation is a single string value, return it.

4.38.3.7 std::string XrslRelation::str ()

Returns a strng representation of the relation.

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

- xrsl.h

4.39 XrslValidationData Class Reference

`#include <xrsl.h>`

Public Member Functions

- **XrslValidationData** (const std::string &**attribute_name**, relation_type **rel_type**, validation_type **val_type**, bool **unique**=true, int **list_length**=0)

Data Fields

- std::string **attribute_name**
- relation_type **rel_type**
- bool **unique**
- int **list_length**
- validation_type **val_type**

4.39.1 Detailed Description

Class for simplifying **Xrsl** (p. 52) validation. One object of this class represents a valid attribute in the xrsl.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 XrslValidationData::XrslValidationData (const std::string & *attribute_name*, relation_type *rel_type*, validation_type *val_type*, bool *unique* = `true`, int *list_length* = `0`)

Constructor.

4.39.3 Field Documentation

4.39.3.1 std::string XrslValidationData::attribute_name

Name of attribute.

4.39.3.2 int XrslValidationData::list_length

Length of each list in case attribute is a list of values.

4.39.3.3 relation_type XrslValidationData::rel_type

Type the attribute must have.

4.39.3.4 bool XrslValidationData::unique

Must this attribute be unique?

4.39.3.5 validation_type XrslValidationData::val_type

Must the **Xrsl** (p. 52) must contain this attribute to be valid?

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

- xrsl.h

Chapter 5

File Documentation

5.1 configcore.h File Reference

```
#include <iostream>
#include <list>
#include <map>
#include <string>
#include <arc/Logger.h>
```

Data Structures

- class **ARex::ConfigError**
- class **ARex::Option**
- class **ARex::ConfGrp**
- class **ARex::Config**

Functions

- Config **ARex::ReadConfig** (std::istream &is)
- Config **ARex::ReadConfig** (const std::string &filename)

5.1.1 Detailed Description

This file describes the core configuration

5.2 PerlProcessor.h File Reference

Header file for the Perl Processor, which implements a procedure to process the incoming tasks.

```
#include <string>
#include <time.h>
#include <arc/Logger.h>
#include "TaskQueue.h"
#include "TaskSet.h"
#include <EXTERN.h>
#include <perl.h>
```

Data Structures

- class **DREService::PerlProcessor**
- struct **DREService::PerlProcessor::ThreadInterface**

5.2.1 Detailed Description

Header file for the Perl Processor, which implements a procedure to process the incoming tasks.

Author:

Michael Glodek

Index