KnowARC Reference Manual

Generated by Doxygen 1.3.5

Wed Apr 4 02:02:58 2007

Contents

Chapter 1

KnowARC Hierarchical Index

1.1 KnowARC Class Hierarchy

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

Chapter 2

KnowARC Class Index

2.1 KnowARC Class List

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

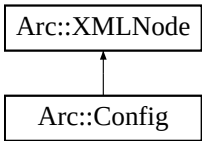
Chapter 3

KnowARC Class Documentation

3.1 Arc::Config Class Reference

`#include <ArcConfig.h>`

Inheritance diagram for Arc::Config::



Public Member Functions

- Config ()
- Config (const char *filename)
- Config (const std::string &xml_str)
- Config (XMLNode xml)
- void print (void)

3.1.1 Detailed Description

Configuration element - represents (sub)tree of ARC configuration. This class is intended to be used to pass configuration details to various parts of HED and external modules. Currently it's just a wrapper over XML tree. But than may change in a future, although interface should be preserved. Currently it is capable of loading XML configuration document from file. In future it will be capable of loading more user-readable format and process it into tree-like structure convenient for machine processing (XML-like). So far there are no schema and/or namespaces assigned.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Arc::Config::Config () `[inline]`

Dummy constructor - produces empty structure

3.1.2.2 Arc::Config::Config (const char * *filename*)

Loads configuration document from file 'filename'

3.1.2.3 Arc::Config::Config (const std::string & *xml_str*) [inline]

Parse configuration document from memory

3.1.2.4 Arc::Config::Config (XMLNode *xml*) [inline]

Acquire existing XML (sub)tree. Content is not copied. Make sure XML tree is not destroyed while in use by this object.

3.1.3 Member Function Documentation

3.1.3.1 void Arc::Config::print (void)

Print structure of document. For debuging purposes. Printed content is not an XML document.

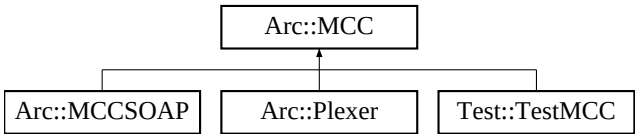
The documentation for this class was generated from the following file:

- ArcConfig.h

3.2 Arc::MCC Class Reference

`#include <MCC.h>`

Inheritance diagram for Arc::MCC::



Public Member Functions

- MCC (Arc::Config *cfg)
- virtual Message process (Message)

3.2.1 Detailed Description

Message Chain Component - base class for every MCC plugin. This is virtual class which defines interface (in a future also common functionality) for every MCC plugin. Interface is made of maethod process() which is called by previous MCC in chain.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Arc::MCC::MCC (Arc::Config * *cfg*) [inline]

Example contructor - MCC takes at least it's configuration subtree

3.2.3 Member Function Documentation

3.2.3.1 virtual Message Arc::MCC::process (Message) [inline, virtual]

Method for processing of information. This method is called by previos MCC in chain. This method must call similar method of next MCC in chain unless any failure happens. Result returned by call to next MCC should be processed and passed back to previous MCC. In case of failure this method is expected to generate valid error response and return it back to previous MCC without calling the next one.

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

- MCC.h

3.3 mcc_descriptor Struct Reference

`#include <MCC.h>`

Public Attributes

- int **version**
- Arc::MCC *(* **get_instance**)(Arc::Config *cfg)

3.3.1 Detailed Description

This structure describes MCC stored in shard library. Each MCC is detected by presence of element named 'descriptor' of this type. It contains version number and pointer to function which creates an instance of object inherited from MCC class.

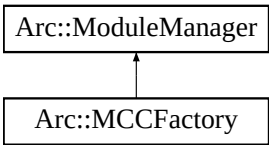
The documentation for this struct was generated from the following file:

- MCC.h

3.4 Arc::MCCFactory Class Reference

`#include <MCCFactory.h>`

Inheritance diagram for Arc::MCCFactory::



Public Member Functions

- MCCFactory (std::string name, Arc::Config *cfg)
- Arc::MCC * get_instance (Arc::Config *cfg)

3.4.1 Detailed Description

This class handles shared libraries containing MCCs

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Arc::MCCFactory::MCCFactory (std::string *name*, Arc::Config * *cfg*)

Constructor - accepts name of module containing MCC and configuration (not used) meant to tune loading of module.

3.4.3 Member Function Documentation

3.4.3.1 Arc::MCC* Arc::MCCFactory::get_instance (Arc::Config * *cfg*)

This method loads shared library named lib'name', locates symbol representing descriptor of MCC and calls it's constructor function. Supplied configuration tree is passed to constructor. Returns created MCC instance.

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

- MCCFactory.h

3.5 Arc::Message Class Reference

`#include <Message.h>`

Public Member Functions

- Message (void)
- MessagePayload * Payload (void)
- MessagePayload * Payload (MessagePayload *new_payload)

3.5.1 Detailed Description

Message is passed through chain of MCCs. It is going to contain main content (payload), authentication/authorization information, attributes, etc.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Arc::Message::Message (void) [inline]

Various useful attributes

3.5.3 Member Function Documentation

3.5.3.1 MessagePayload* Arc::Message::Payload (MessagePayload * *new_payload*) [inline]

Replace payload with new one

3.5.3.2 MessagePayload* Arc::Message::Payload (void) [inline]

Returns pointer to current payload or NULL if no payload assigned.

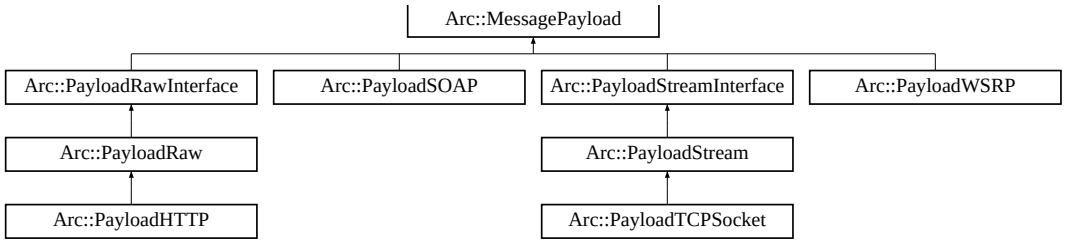
The documentation for this class was generated from the following file:

- Message.h

3.6 Arc::MessagePayload Class Reference

`#include <Message.h>`

Inheritance diagram for Arc::MessagePayload::



3.6.1 Detailed Description

Base class for content of message passed through chain. It's not intended to be used directly. Instead functional classes must be derived from it.

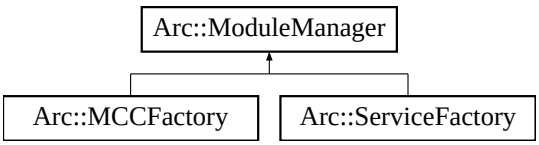
The documentation for this class was generated from the following file:

- Message.h

3.7 Arc::ModuleManager Class Reference

`#include <ModuleManager.h>`

Inheritance diagram for Arc::ModuleManager::



Public Member Functions

- ModuleManager (Arc::Config *cfg)
- Glib::Module * load (const std::string &name)

3.7.1 Detailed Description

This class loads shared libraries/modules. There supposed to be created one instance of it per executable. In such circumstances it would cache handles to loaded modules and not load them multiple times.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Arc::ModuleManager::ModuleManager (Arc::Config * *cfg*)

Constructor. It is supposed to process correponding configuration subtree and tune module loading parameters accordingly. Currently it only sets modulr directory to current one.

3.7.3 Member Function Documentation

3.7.3.1 Glib::Module* Arc::ModuleManager::load (const std::string & *name*)

Finds module 'name' in cache or loads corresponding shared library

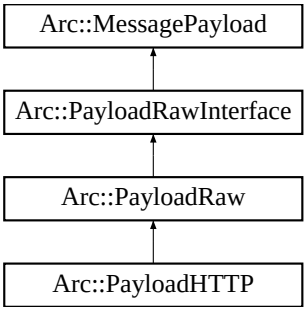
The documentation for this class was generated from the following file:

- ModuleManager.h

3.8 Arc::PayloadHTTP Class Reference

`#include <PayloadHTTP.h>`

Inheritance diagram for Arc::PayloadHTTP::



Public Member Functions

- PayloadHTTP (PayloadStreamInterface &stream)
- PayloadHTTP (const std::string &method, const std::string &url, PayloadStreamInterface &stream)
- PayloadHTTP (int code, const std::string &reason, PayloadStreamInterface &stream)
- virtual const std::string & Attribute (const std::string &name)
- virtual void Attribute (const std::string &name, const std::string &value)
- virtual bool Flush (void)

Protected Member Functions

- bool readline (std::string &line)
- bool read (char *buf, int &size)
- bool parse_header (void)
- bool get_body (void)

Protected Attributes

- PayloadStreamInterface & **stream_**
- std::string uri_
- int version_major_
- int version_minor_
- std::string method_
- int code_
- std::string reason_
- int length_
- bool chunked_
- std::map< std::string, std::string > attributes_
- char **tbuf_** [1024]
- int **tbuflen_**

3.8.1 Detailed Description

This class implements parsing and generation of HTTP messages. It implements only subset of HTTP/1.1 and also provides an PayloadRawInterface for including as payload into Message passed through MCC chains.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Arc::PayloadHTTP::PayloadHTTP (PayloadStreamInterface & *stream*)

Constructor - creates object by parsing HTTP request or response from stream. Supplied stream is associated with object for later use.

3.8.2.2 Arc::PayloadHTTP::PayloadHTTP (const std::string & *method*, const std::string & *url*, PayloadStreamInterface & *stream*)

Constructor - creates HTTP request to be sent through stream. HTTP message is not sent yet.

3.8.2.3 Arc::PayloadHTTP::PayloadHTTP (int *code*, const std::string & *reason*, PayloadStreamInterface & *stream*)

Constructor - creates HTTP response to be sent through stream. HTTP message is not sent yet.

3.8.3 Member Function Documentation

3.8.3.1 virtual void Arc::PayloadHTTP::Attribute (const std::string & *name*, const std::string & *value*) [virtual]

Sets HTTP header attribute 'name' to 'value'

3.8.3.2 virtual const std::string& Arc::PayloadHTTP::Attribute (const std::string & *name*) [virtual]

Returns HTTP header attribute with specified name. Empty string if nosuch attribute.

3.8.3.3 virtual bool Arc::PayloadHTTP::Flush (void) [virtual]

Send created object through associated stream

3.8.3.4 bool Arc::PayloadHTTP::get_body (void) [protected]

Read Body of HTTP message and attach it to inherited PayloadRaw object

3.8.3.5 bool Arc::PayloadHTTP::parse_header (void) [protected]

Read HTTP header and fill internal variables

3.8.3.6 bool Arc::PayloadHTTP::read (char * *buf*, int & *size*) [protected]

Read up to 'size' bytes from stream_

3.8.3.7 bool Arc::PayloadHTTP::readline (std::string & *line*) [protected]

Read from stream till

3.8.4 Member Data Documentation

3.8.4.1 std::map<std::string,std::string> Arc::PayloadHTTP::attributes_ [protected]

true if content is chunked

3.8.4.2 bool Arc::PayloadHTTP::chunked_ [protected]

Content-length of HTTP message

3.8.4.3 int Arc::PayloadHTTP::code_ [protected]

HTTP method being used or requested

3.8.4.4 int Arc::PayloadHTTP::length_ [protected]

HTTP reason being sent or supplied

3.8.4.5 std::string Arc::PayloadHTTP::method_ [protected]

minor number of HTTP version - must be 0 or 1

3.8.4.6 std::string Arc::PayloadHTTP::reason_ [protected]

HTTP code being sent or supplied

3.8.4.7 std::string Arc::PayloadHTTP::uri_ [protected]

stream used to comminicate to outside

3.8.4.8 int Arc::PayloadHTTP::version_major_ [protected]

URI being contacted

3.8.4.9 int Arc::PayloadHTTP::version_minor_ [protected]

major number of HTTP version - must be 1

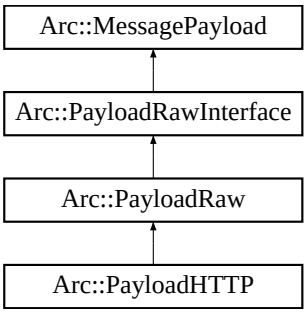
The documentation for this class was generated from the following file:

- PayloadHTTP.h

3.9 Arc::PayloadRaw Class Reference

`#include <PayloadRaw.h>`

Inheritance diagram for Arc::PayloadRaw::



Public Member Functions

- PayloadRaw (void)
- virtual ~PayloadRaw (void)
- virtual char operator[ ] (int pos) const
- virtual char * Content (int pos=-1)
- virtual int Size (void) const
- virtual char * Insert (int pos=0, int size=0)
- virtual char * Insert (const char *s, int pos=0, int size=0)
- virtual char * Buffer (int num=0)
- virtual int BufferSize (int num=0) const

Protected Attributes

- std::vector< Buf > **buf_**

3.9.1 Detailed Description

Implementation of PayloadRawInterface - raw byte multi-buffer.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Arc::PayloadRaw::PayloadRaw (void) `[inline]`

Constructor. Created object contains no buffers.

3.9.2.2 virtual Arc::PayloadRaw::~PayloadRaw (void) `[virtual]`

Destructor. Frees allocated buffers.

3.9.3 Member Function Documentation

3.9.3.1 virtual char* Arc::PayloadRaw::Buffer (int *num* = 0) [virtual]

Returns pointer to num'th buffer

Implements Arc::PayloadRawInterface.

3.9.3.2 virtual int Arc::PayloadRaw::BufferSize (int *num* = 0) const [virtual]

Returns length of num'th buffer

Implements Arc::PayloadRawInterface.

3.9.3.3 virtual char* Arc::PayloadRaw::Content (int *pos* = -1) [virtual]

Get pointer to buffer content at global position 'pos'. By default to beginning of main buffer whatever that means.

Implements Arc::PayloadRawInterface.

3.9.3.4 virtual char* Arc::PayloadRaw::Insert (const char * *s*, int *pos* = 0, int *size* = 0) [virtual]

Create new buffer at global position 'pos' of size 'size'. Created buffer is filled with content of memory at 's'. If 'size' is 0 aontent at 's' is expected to be null-terminated.

Implements Arc::PayloadRawInterface.

3.9.3.5 virtual char* Arc::PayloadRaw::Insert (int *pos* = 0, int *size* = 0) [virtual]

Create new buffer at global position 'pos' of size 'size'.

Implements Arc::PayloadRawInterface.

3.9.3.6]

virtual char Arc::PayloadRaw::operator[ ] (int *pos*) const [virtual]

Returns content of byte at specified position. Specified position 'pos' is treated as global one and goes through all buffers placed one after another.

Implements Arc::PayloadRawInterface.

3.9.3.7 virtual int Arc::PayloadRaw::Size (void) const [virtual]

Returns cumulative length of all buffers

Implements Arc::PayloadRawInterface.

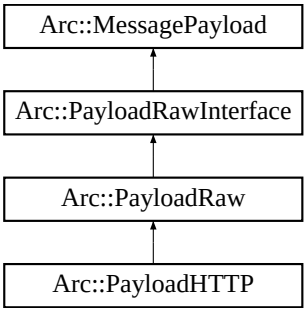
The documentation for this class was generated from the following file:

- PayloadRaw.h

3.10 Arc::PayloadRawInterface Class Reference

`#include <PayloadRaw.h>`

Inheritance diagram for Arc::PayloadRawInterface::



Public Member Functions

- virtual char operator[ ] (int pos) const=0
- virtual char * Content (int pos=-1)=0
- virtual int Size (void) const=0
- virtual char * Insert (int pos=0, int size=0)=0
- virtual char * Insert (const char *s, int pos=0, int size=0)=0
- virtual char * Buffer (int num)=0
- virtual int BufferSize (int num) const=0

3.10.1 Detailed Description

Virtual interface for managing arbitrarily accessible Message payload. This class implements memory-resident or memory-mapped content made of optionally multiple chunks/buffers. This calss is purely virtual.

3.10.2 Member Function Documentation

3.10.2.1 virtual char* Arc::PayloadRawInterface::Buffer (int *num*) [pure virtual]

Returns pointer to num'th buffer

Implemented in Arc::PayloadRaw.

3.10.2.2 virtual int Arc::PayloadRawInterface::BufferSize (int *num*) const [pure virtual]

Returns length of num'th buffer

Implemented in Arc::PayloadRaw.

3.10.2.3 virtual char* Arc::PayloadRawInterface::Content (int *pos* = -1) [pure virtual]

Get pointer to buffer content at global position 'pos'. By default to beginning of main buffer whatever that means.

Implemented in Arc::PayloadRaw.

3.10.2.4 virtual char* Arc::PayloadRawInterface::Insert (const char * *s*, int *pos* = 0, int *size* = 0) [pure virtual]

Create new buffer at global position 'pos' of size 'size'. Created buffer is filled with content of memory at 's'. If 'size' is 0 aontent at 's' is expected to be null-terminated.

Implemented in Arc::PayloadRaw.

3.10.2.5 virtual char* Arc::PayloadRawInterface::Insert (int *pos* = 0, int *size* = 0) [pure virtual]

Create new buffer at global position 'pos' of size 'size'.

Implemented in Arc::PayloadRaw.

3.10.2.6]

virtual char Arc::PayloadRawInterface::operator[ ] (int *pos*) const [pure virtual]

Returns content of byte at specified position. Specified position 'pos' is treated as global one and goes through all buffers placed one after another.

Implemented in Arc::PayloadRaw.

3.10.2.7 virtual int Arc::PayloadRawInterface::Size (void) const [pure virtual]

Returns cumulative length of all buffers

Implemented in Arc::PayloadRaw.

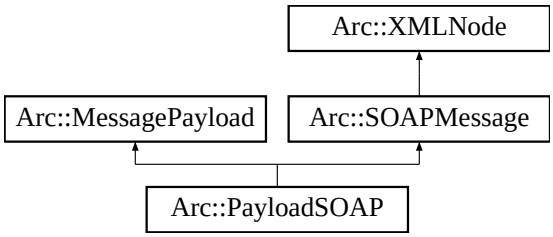
The documentation for this class was generated from the following file:

- PayloadRaw.h

3.11 Arc::PayloadSOAP Class Reference

`#include <PayloadSOAP.h>`

Inheritance diagram for Arc::PayloadSOAP::



Public Member Functions

- PayloadSOAP (const SOAPMessage &soap)
- PayloadSOAP (const MessagePayload &source)

3.11.1 Detailed Description

This class combines MessagePayload with SOAPMessage to make it possible to pass SOAP messages through MCC chain

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Arc::PayloadSOAP::PayloadSOAP (const SOAPMessage & *soap*)

Constructor - creates Message payload from SOAP message

3.11.2.2 Arc::PayloadSOAP::PayloadSOAP (const MessagePayload & *source*)

Constructor - creates SOAP message from payload. PayloadRawInterface and derived classes are supported.

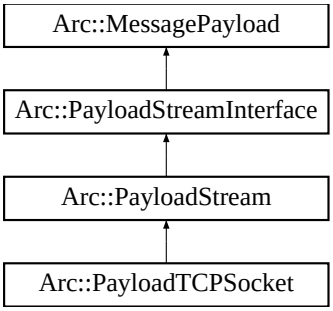
The documentation for this class was generated from the following file:

- PayloadSOAP.h

3.12 Arc::PayloadStream Class Reference

`#include <PayloadStream.h>`

Inheritance diagram for Arc::PayloadStream::



Public Member Functions

- PayloadStream (int h=-1)
- ~PayloadStream (void)
- virtual bool Get (char *buf, int &size)
- virtual bool Get (std::string &buf)
- virtual std::string Get (void)
- virtual bool Put (const char *buf, int size)
- virtual bool Put (const std::string &buf)
- virtual bool Put (const char *buf)
- virtual operator bool (void)
- virtual bool operator! (void)
- virtual int Timeout (void) const
- virtual void Timeout (int to)

Protected Attributes

- int **timeout_**
- int handle_
- bool seekable_

3.12.1 Detailed Description

Implemetation of PayloadStreamInterface for generic POSIX handle.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 Arc::PayloadStream::PayloadStream (int *h* = -1)

Constructor. Takes already open handle as argument.

3.12.2.2 Arc::PayloadStream::~PayloadStream (void) [inline]

Destructor. Closes acquired handle.

3.12.3 Member Function Documentation

3.12.3.1 virtual std::string Arc::PayloadStream::Get (void) [inline, virtual]

Read as many as possible (sane amount) of bytes.

Implements Arc::PayloadStreamInterface.

3.12.3.2 virtual bool Arc::PayloadStream::Get (std::string & *buf*) [virtual]

Read as many as possible (sane amount) of bytes into buf.

Implements Arc::PayloadStreamInterface.

3.12.3.3 virtual bool Arc::PayloadStream::Get (char * *buf*, int & *size*) [virtual]

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implements Arc::PayloadStreamInterface.

3.12.3.4 virtual Arc::PayloadStream::operator bool (void) [inline, virtual]

Returns true if stream is valid.

Implements Arc::PayloadStreamInterface.

3.12.3.5 virtual bool Arc::PayloadStream::operator! (void) [inline, virtual]

Returns true if stream is invalid.

Implements Arc::PayloadStreamInterface.

3.12.3.6 virtual bool Arc::PayloadStream::Put (const char * *buf*) [inline, virtual]

Push null terminated information from 'buf' into stream. Returns true on success.

Implements Arc::PayloadStreamInterface.

3.12.3.7 virtual bool Arc::PayloadStream::Put (const std::string & *buf*) [inline, virtual]

Push information from 'buf' into stream. Returns true on success.

Implements Arc::PayloadStreamInterface.

3.12.3.8 virtual bool Arc::PayloadStream::Put (const char * *buf*, int *size*) [virtual]

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implements Arc::PayloadStreamInterface.

3.12.3.9 virtual void Arc::PayloadStream::Timeout (int *to*) [inline, virtual]

Set current timeout for Get() and Put() operations.

Implements Arc::PayloadStreamInterface.

3.12.3.10 virtual int Arc::PayloadStream::Timeout (void) const [inline, virtual]

Query current timeout for Get() and Put() operations.

Implements Arc::PayloadStreamInterface.

3.12.4 Member Data Documentation

3.12.4.1 int Arc::PayloadStream::handle_ [protected]

Timeout for read/write operations

3.12.4.2 bool Arc::PayloadStream::seekable_ [protected]

Handle for operations

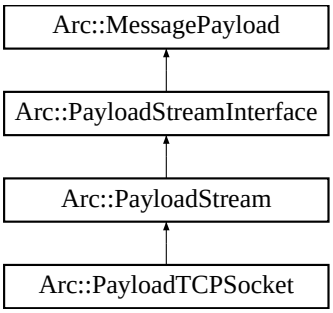
The documentation for this class was generated from the following file:

- PayloadStream.h

3.13 Arc::PayloadStreamInterface Class Reference

`#include <PayloadStream.h>`

Inheritance diagram for Arc::PayloadStreamInterface::



Public Member Functions

- virtual bool Get (char *buf, int &size)=0
- virtual bool Get (std::string &buf)=0
- virtual std::string Get (void)=0
- virtual bool Put (const char *buf, int size)=0
- virtual bool Put (const std::string &buf)=0
- virtual bool Put (const char *buf)=0
- virtual operator bool (void)=0
- virtual bool operator! (void)=0
- virtual int Timeout (void) const=0
- virtual void Timeout (int to)=0

3.13.1 Detailed Description

Virtual interface for managing stream-like source and destination. It's supposed to passed through MCC chain as payload of Message. It must be treated by MCC as dynamic payload. This class is purely virtual.

3.13.2 Member Function Documentation

3.13.2.1 virtual std::string Arc::PayloadStreamInterface::Get (void) [pure virtual]

Read as many as possible (sane amount) of bytes.

Implemented in Arc::PayloadStream.

3.13.2.2 virtual bool Arc::PayloadStreamInterface::Get (std::string & *buf*) [pure virtual]

Read as many as possible (sane amount) of bytes into buf.

Implemented in Arc::PayloadStream.

3.13.2.3 virtual bool Arc::PayloadStreamInterface::Get (char * *buf*, int & *size*) [pure virtual]

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implemented in Arc::PayloadStream.

3.13.2.4 virtual Arc::PayloadStreamInterface::operator bool (void) [pure virtual]

Returns true if stream is valid.

Implemented in Arc::PayloadStream.

3.13.2.5 virtual bool Arc::PayloadStreamInterface::operator! (void) [pure virtual]

Returns true if stream is invalid.

Implemented in Arc::PayloadStream.

3.13.2.6 virtual bool Arc::PayloadStreamInterface::Put (const char * *buf*) [pure virtual]

Push null terminated information from 'buf' into stream. Returns true on success.

Implemented in Arc::PayloadStream.

3.13.2.7 virtual bool Arc::PayloadStreamInterface::Put (const std::string & *buf*) [pure virtual]

Push information from 'buf' into stream. Returns true on success.

Implemented in Arc::PayloadStream.

3.13.2.8 virtual bool Arc::PayloadStreamInterface::Put (const char * *buf*, int *size*) [pure virtual]

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implemented in Arc::PayloadStream.

3.13.2.9 virtual void Arc::PayloadStreamInterface::Timeout (int *to*) [pure virtual]

Set current timeout for Get() and Put() operations.

Implemented in Arc::PayloadStream.

3.13.2.10 virtual int Arc::PayloadStreamInterface::Timeout (void) const [pure virtual]

Query current timeout for Get() and Put() operations.

Implemented in Arc::PayloadStream.

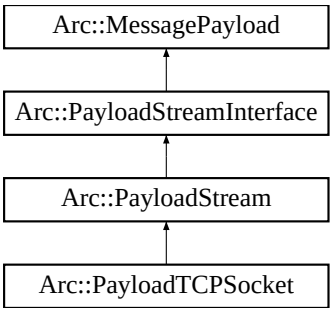
The documentation for this class was generated from the following file:

- PayloadStream.h

3.14 Arc::PayloadTCPSocket Class Reference

`#include <PayloadTCPSocket.h>`

Inheritance diagram for Arc::PayloadTCPSocket::



Public Member Functions

- PayloadTCPSocket (const char *hostname, int port)
- PayloadTCPSocket (const std::string endpoint)
- PayloadTCPSocket (int s)

3.14.1 Detailed Description

This class extends PayloadStream with TCP socket specific features

3.14.2 Constructor & Destructor Documentation

3.14.2.1 Arc::PayloadTCPSocket::PayloadTCPSocket (const char * *hostname*, int *port*)

Constructor - connects to TCP server at specified hostname:port

3.14.2.2 Arc::PayloadTCPSocket::PayloadTCPSocket (const std::string *endpoint*)

Constructor - conencts to RCP server at specified endpoint - hostname:port

3.14.2.3 Arc::PayloadTCPSocket::PayloadTCPSocket (int *s*) `[inline]`

Constructor - creates object of already connected socket

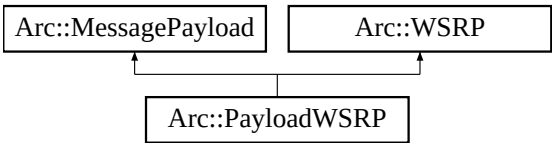
The documentation for this class was generated from the following file:

- PayloadTCPSocket.h

3.15 Arc::PayloadWSRP Class Reference

`#include <PayloadWSRP.h>`

Inheritance diagram for Arc::PayloadWSRP::



Public Member Functions

- PayloadWSRP (const SOAPMessage &soap)
- PayloadSOAP (const MessagePayload &source)

3.15.1 Detailed Description

This class combines MessagePayload with WSRP to make it possible to pass WS-ResourceProperties messages through MCC chain

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Arc::PayloadWSRP::PayloadWSRP (const SOAPMessage & *soap*)

Constructor - creates Message payload from SOAP message Returns invalid WSRP if SOAP does not represent WS-ResourceProperties

3.15.3 Member Function Documentation

3.15.3.1 Arc::PayloadWSRP::PayloadSOAP (const MessagePayload & *source*)

Constructor - creates Message payload from WSRP message PayloadWSRP(const WSRP& wsrp); /** Constructor - creates WSRP message from payload. All classes derived from SOAPMessage are supported.

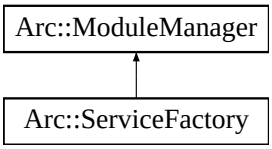
The documentation for this class was generated from the following file:

- PayloadWSRP.h

3.16 Arc::ServiceFactory Class Reference

`#include <ServiceFactory.h>`

Inheritance diagram for Arc::ServiceFactory::



Public Member Functions

- ServiceFactory (std::string name, Arc::Config *cfg)
- Arc::Service * get_instance (Arc::Config *cfg)

3.16.1 Detailed Description

This class handles shared libraries containing Services

3.16.2 Constructor & Destructor Documentation

3.16.2.1 Arc::ServiceFactory::ServiceFactory (std::string *name*, Arc::Config * *cfg*)

Constructor - accepts name of module containing Service and configuration (not used) meant to tune loading of module.

3.16.3 Member Function Documentation

3.16.3.1 Arc::Service* Arc::ServiceFactory::get_instance (Arc::Config * *cfg*)

This method loads shared library named lib'name', locates symbol representing descriptor of Service and calls it's constructor function. Supplied configuration tree is passed to constructor. Returns created Service instance.

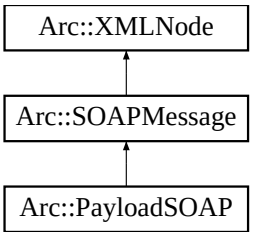
The documentation for this class was generated from the following file:

- ServiceFactory.h

3.17 Arc::SOAPMessage Class Reference

`#include <SOAPMessage.h>`

Inheritance diagram for Arc::SOAPMessage::



Public Member Functions

- SOAPMessage (const std::string &xml)
- SOAPMessage (const char *xml, int len=-1)
- SOAPMessage (const NS &ns, bool fault=false)
- void Namespaces (const NS &namespaces)
- void GetXML (std::string &xml) const
- XMLNode **Header** (void)
- bool **IsFault** (void)
- SOAPFault * **Fault** (void)

3.17.1 Detailed Description

SOAPMessage extends XMLNode class to support structures of SOAP message. All XMLNode methods are exposed with top node translated to Envelope part of SOAP.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 Arc::SOAPMessage::SOAPMessage (const std::string & *xml*)

Create new SOAP message from textual representation of XML document. Created XML structure is owned by this instance. This constructor also sets default namespaces to default prefixes as specified below.

3.17.2.2 Arc::SOAPMessage::SOAPMessage (const char * *xml*, int *len* = -1)

Same as previous

3.17.2.3 Arc::SOAPMessage::SOAPMessage (const NS & *ns*, bool *fault* = false)

Create new SOAP message with specified namespaces. Created XML structure is owned by this instance. If argument fault is set to true created message is fault.

3.17.3 Member Function Documentation

3.17.3.1 void Arc::SOAPMessage::GetXML (std::string & *xml*) const

Fills argument with this instance XML (sub)tree textual representation

Reimplemented from Arc::XMLNode.

3.17.3.2 void Arc::SOAPMessage::Namespaces (const NS & *namespaces*)

Modify assigned namespaces. Default namespaces and prefixes are soap-enc http://schemas.xmlsoap.org/soap/encoding/ soap-env http://schemas.xmlsoap.org/soap/envelope/ xsi http://www.w3.org/2001/XMLSchema-instance xsd http://www.w3.org/2001/XMLSchema

Reimplemented from Arc::XMLNode.

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

- SOAPMessage.h

3.18 Arc::SOAPMessage::SOAPFault Class Reference

`#include <SOAPMessage.h>`

Public Types

- enum SOAPFaultCode {

  undefined, **unknown**, **VersionMismatch**, **MustUnderstand**,

  Sender, **Receiver**, **DataEncodingUnknown** }

Public Member Functions

- SOAPFault (XMLNode &body)
- operator bool (void)
- SOAPFaultCode Code (void)
- void Code (SOAPFaultCode code)
- std::string Subcode (int level)
- void Subcode (int level, const char *subcode)
- std::string Reason (int num=0)
- void Reason (int num, const char *reason)
- void Reason (const char *reason)
- std::string Node (void)
- void Node (const char *node)
- std::string Role (void)
- void Role (const char *role)
- XMLNode Detail (bool create=false)

Friends

- class **SOAPMessage**

3.18.1 Detailed Description

SOAPFault provides an interface to conveinet access to elements of SOAP faults. It also tries to expose single interface for both version 1.0 and 1.2 faults. This class is not intended to 'own' any information stored. It's purpose is to manipulate information which under control of XMLNode or SOAPMessage classes. If instance does not refer to valid SOAP Fault structure all manipulation methods will have no effect.

3.18.2 Member Enumeration Documentation

3.18.2.1 enum Arc::SOAPMessage::SOAPFault::SOAPFaultCode

Fault codes of SOAP specs

3.18.3 Constructor & Destructor Documentation

3.18.3.1 Arc::SOAPMessage::SOAPFault::SOAPFault (XMLNode & *body*)

Parse Fault elements of SOAP Body or any other XML tree with Fault element

3.18.4 Member Function Documentation

3.18.4.1 void Arc::SOAPMessage::SOAPFault::Code (SOAPFaultCode *code*)

Set Fault Code element

3.18.4.2 SOAPFaultCode Arc::SOAPMessage::SOAPFault::Code (void)

Returns Fault Code element

3.18.4.3 XMLNode Arc::SOAPMessage::SOAPFault::Detail (bool *create* = false)

Access Fault Detail element. If create is set to true this element is creted if not present.

3.18.4.4 void Arc::SOAPMessage::SOAPFault::Node (const char * *node*)

Set content of Fault Node element to 'node'

3.18.4.5 std::string Arc::SOAPMessage::SOAPFault::Node (void)

Returns content of Fault Node element

3.18.4.6 Arc::SOAPMessage::SOAPFault::operator bool (void) [inline]

Returns true if instance refers to SOAP Fault

3.18.4.7 void Arc::SOAPMessage::SOAPFault::Reason (const char * *reason*) [inline]

Set Fault Reason element at top level

3.18.4.8 void Arc::SOAPMessage::SOAPFault::Reason (int *num*, const char * *reason*)

Set Fault Reason content at various levels to 'reason'

3.18.4.9 std::string Arc::SOAPMessage::SOAPFault::Reason (int *num* = 0)

Returns content of Fault Reason element at various levels

3.18.4.10 void Arc::SOAPMessage::SOAPFault::Role (const char * *role*)

Set content of Fault Role element to 'role'

3.18.4.11 std::string Arc::SOAPMessage::SOAPFault::Role (void)

Returns content of Fault Role element

3.18.4.12 void Arc::SOAPMessage::SOAPFault::Subcode (int *level*, const char * *subcode*)

Set Fault Subcode element at various levels (0 is for Code) to 'subcode'

3.18.4.13 std::string Arc::SOAPMessage::SOAPFault::Subcode (int *level*)

Returns Fault Subcode element at various levels (0 is for Code)

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

- SOAPMessage.h

3.19 Arc::WSAEndpointReference Class Reference

`#include <WSA.h>`

Public Member Functions

- WSAEndpointReference (XMLNode epr)
- WSAEndpointReference (const std::string &address)
- WSAEndpointReference (void)
- ~WSAEndpointReference (void)
- std::string Address (void) const
- void Address (const std::string &uri)
- WSAEndpointReference & operator= (const std::string &address)
- XMLNode ReferenceParameters (void)
- XMLNode MetaData (void)
- operator XMLNode (void)

Protected Attributes

- XMLNode **epr_**

3.19.1 Detailed Description

This class implements interface for manipulating WS-Adressing Endpoint Reference stored in XML tree. Question: should there be some standalone class for storing EPR information?

3.19.2 Constructor & Destructor Documentation

3.19.2.1 Arc::WSAEndpointReference::WSAEndpointReference (XMLNode *epr*)

Linking to existing EPR in XML tree

3.19.2.2 Arc::WSAEndpointReference::WSAEndpointReference (const std::string & *address*)

Creating independent EPR - not implemented

3.19.2.3 Arc::WSAEndpointReference::WSAEndpointReference (void)

Dummy constructor - creates invalid instance

3.19.2.4 Arc::WSAEndpointReference::~WSAEndpointReference (void)

Destructor. All empty elements of EPR XML are destroyed here too

3.19.3 Member Function Documentation

3.19.3.1 void Arc::WSAEndpointReference::Address (const std::string & *uri*)

Assigns new Address value. If EPR had no Address element it is created.

3.19.3.2 std::string Arc::WSAEndpointReference::Address (void) const

Returns Address (URL) encoded in EPR

3.19.3.3 XMLNode Arc::WSAEndpointReference::MetaData (void)

Access to MetaData element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no MetaData element it is created.

3.19.3.4 Arc::WSAEndpointReference::operator XMLNode (void)

Returns reference to EPR top XML node

3.19.3.5 WSAEndpointReference& Arc::WSAEndpointReference::operator= (const std::string & *address*)

Same as Address(uri)

3.19.3.6 XMLNode Arc::WSAEndpointReference::ReferenceParameters (void)

Access to ReferenceParameters element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no ReferenceParameters element it is created.

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

- WSA.h

3.20 Arc::WSAHeader Class Reference

`#include <WSA.h>`

Public Member Functions

- WSAHeader (SOAPMessage &soap)
- WSAHeader (const std::string &action)
- std::string To (void) const
- void To (const std::string &uri)
- WSAEndpointReference From (void)
- WSAEndpointReference ReplyTo (void)
- WSAEndpointReference FaultTo (void)
- std::string Action (void) const
- void Action (const std::string &uri)
- std::string MessageID (void) const
- void MessageID (const std::string &uri)
- std::string RelatesTo (void) const
- void RelatesTo (const std::string &uri)
- std::string RelationshipType (void) const
- void RelationshipType (const std::string &uri)
- XMLNode ReferenceParameter (int n)
- XMLNode ReferenceParameter (const std::string &name)
- XMLNode NewReferenceParameter (const std::string &name)
- operator XMLNode (void)

Protected Attributes

- XMLNode **header_**
- bool header_allocated_

3.20.1 Detailed Description

Interface to manipulate WS-Addressing related information in SOAP header

3.20.2 Constructor & Destructor Documentation

3.20.2.1 Arc::WSAHeader::WSAHeader (SOAPMessage & *soap*)

Linking to a header of existing SOAP message

3.20.2.2 Arc::WSAHeader::WSAHeader (const std::string & *action*)

Creating independent SOAP header - not implemented

3.20.3 Member Function Documentation

3.20.3.1 void Arc::WSAHeader::Action (const std::string & *uri*)

Set content of Action element of SOAP Header. If such element does not exist it's created.

3.20.3.2 std::string Arc::WSAHeader::Action (void) const

Returns content of Action element of SOAP Header.

3.20.3.3 WSAEndpointReference Arc::WSAHeader::FaultTo (void)

Returns FaultTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

3.20.3.4 WSAEndpointReference Arc::WSAHeader::From (void)

Returns From element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

3.20.3.5 void Arc::WSAHeader::MessageID (const std::string & *uri*)

Set content of MessageID element of SOAP Header. If such element does not exist it's created.

3.20.3.6 std::string Arc::WSAHeader::MessageID (void) const

Returns content of MessageID element of SOAP Header.

3.20.3.7 XMLNode Arc::WSAHeader::NewReferenceParameter (const std::string & *name*)

Creates new ReferenceParameter element with specified name. Returns reference to created element.

3.20.3.8 Arc::WSAHeader::operator XMLNode (void)

Returns reference to SOAP Header - not implemented

3.20.3.9 XMLNode Arc::WSAHeader::ReferenceParameter (const std::string & *name*)

Returns first ReferenceParameter element with specified name

3.20.3.10 XMLNode Arc::WSAHeader::ReferenceParameter (int *n*)

Return n-th ReferenceParameter element

3.20.3.11 void Arc::WSAHeader::RelatesTo (const std::string & *uri*)

Set content of RelatesTo element of SOAP Header. If such element does not exist it's created.

3.20.3.12 std::string Arc::WSAHeader::RelatesTo (void) const

Returns content of RelatesTo element of SOAP Header.

3.20.3.13 void Arc::WSAHeader::RelationshipType (const std::string & *uri*)

Set content of RelationshipType element of SOAP Header. If such element does not exist it's created.

3.20.3.14 std::string Arc::WSAHeader::RelationshipType (void) const

Returns content of RelationshipType element of SOAP Header.

3.20.3.15 WSAEndpointReference Arc::WSAHeader::ReplyTo (void)

Returns ReplyTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulted.

3.20.3.16 void Arc::WSAHeader::To (const std::string & *uri*)

Set content of To element of SOAP Header. If such element does not exist it's created.

3.20.3.17 std::string Arc::WSAHeader::To (void) const

Returns content of To element of SOAP Header.

3.20.4 Member Data Documentation

3.20.4.1 bool Arc::WSAHeader::header_allocated_ [protected]

SOAP header element

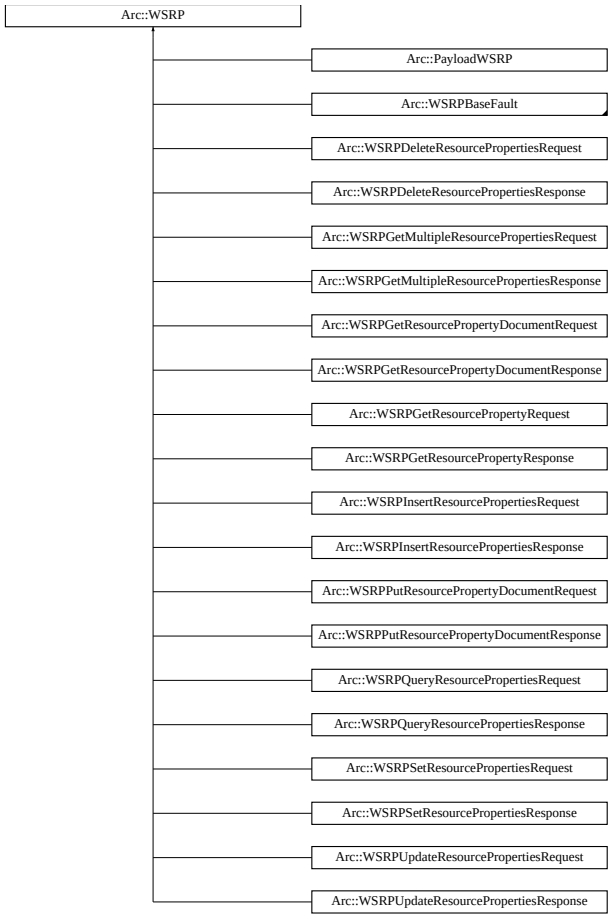
The documentation for this class was generated from the following file:

- WSA.h

3.21 Arc::WSRP Class Reference

`#include <WSResourceProperties.h>`

Inheritance diagram for Arc::WSRP::



Public Member Functions

- WSRP (bool fault=false, const std::string &action="")
- WSRP (SOAPMessage &soap, const std::string &action="")
- operator bool (void)
- SOAPMessage & SOAP (void)

Protected Member Functions

- void set_namespaces (void)

Protected Attributes

- SOAPMessage & **soap_**
- bool allocated_

- bool valid_

3.21.1 Detailed Description

Base class for all WS-ResourceProperties structures. Inheriting classes implement specific WS-ResourceProperties messages and their properties/elements. Refer to WS-ResourceProperties specifications for things specific to every message.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Arc::WSRP::WSRP (bool *fault* = false, const std::string & *action* = "")

Constructor - prepares object for creation of new WSRP request/response/fault

3.21.2.2 Arc::WSRP::WSRP (SOAPMessage & *soap*, const std::string & *action* = "")

Constructor - creates object out of supplied SOAP tree. It does not check if 'soap' represents valid WS-ResourceProperties structure. Actual check for validity of structure has to be done by derived class.

3.21.3 Member Function Documentation

3.21.3.1 Arc::WSRP::operator bool (void) [inline]

Returns true if instance is valid

3.21.3.2 void Arc::WSRP::set_namespaces (void) [protected]

set WS-ResourceProperties namespaces and default prefixes in SOAP message

3.21.3.3 SOAPMessage& Arc::WSRP::SOAP (void) [inline]

Direct access to underlying SOAP element

3.21.4 Member Data Documentation

3.21.4.1 bool Arc::WSRP::allocated_ [protected]

Associated SOAP message

3.21.4.2 bool Arc::WSRP::valid_ [protected]

true if soap_ needs to be deleted in destructor

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

- WSResourceProperties.h

3.22 Arc::WSRPBaseFault Class Reference

`#include <WSResourceProperties.h>`

Inheritance diagram for Arc::WSRPBaseFault::

Public Member Functions

- WSRPBaseFault (SOAPMessage &soap)
- WSRPBaseFault (void)

3.22.1 Detailed Description

Base class for all WS-ResourceProperties faults

3.22.2 Constructor & Destructor Documentation

3.22.2.1 Arc::WSRPBaseFault::WSRPBaseFault (SOAPMessage & *soap*)

Constructor - creates object out of supplied SOAP tree.

3.22.2.2 Arc::WSRPBaseFault::WSRPBaseFault (void)

Constructor - creates new WSRP fault

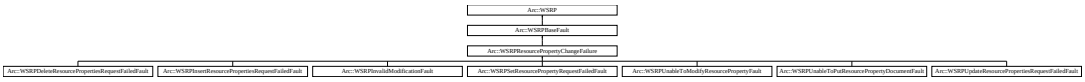
The documentation for this class was generated from the following file:

- WSResourceProperties.h

3.23 Arc::WSRPResourcePropertyChangeFailure Class Reference

`#include <WSResourceProperties.h>`

Inheritance diagram for Arc::WSRPResourcePropertyChangeFailure::



Public Member Functions

- WSRPResourcePropertyChangeFailure (SOAPMessage &soap)
- WSRPResourcePropertyChangeFailure (void)
- XMLNode **CurrentProperties** (bool create=false)
- XMLNode **RequestedProperties** (bool create=false)

3.23.1 Detailed Description

Base class for WS-ResourceProperties faults which contain ResourcePropertyChangeFailure

3.23.2 Constructor & Destructor Documentation

3.23.2.1 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (SOAPMessage & *soap*) [inline]

Constructor - creates object out of supplied SOAP tree.

3.23.2.2 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (void) [inline]

Constructor - creates new WSRP fault

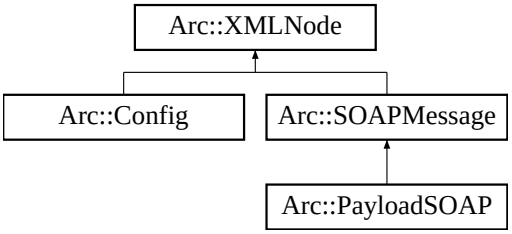
The documentation for this class was generated from the following file:

- WSResourceProperties.h

3.24 Arc::XMLNode Class Reference

`#include <XMLNode.h>`

Inheritance diagram for Arc::XMLNode::



Public Types

- typedef std::map< std::string, std::string > NS

Public Member Functions

- XMLNode (void)
- XMLNode (const XMLNode &node)
- XMLNode (const std::string &xml)
- XMLNode (const char *xml, int len=-1)
- XMLNode (const NS &ns)
- ~XMLNode (void)
- operator bool (void) const
- bool operator! (void) const
- XMLNode Child (int n=0) const
- XMLNode operator[ ] (const char *name) const
- XMLNode operator[ ] (const std::string &name) const
- XMLNode operator[ ] (int n) const
- int Size (void) const
- std::string Name (void) const
- void Name (std::string name)
- void GetXML (std::string &xml) const
- operator std::string (void) const
- XMLNode & operator= (const std::string &content)
- XMLNode & operator= (const char *content)
- XMLNode & operator= (const XMLNode &node)
- std::list< XMLNode > Attributes (void)
- XMLNode Attribute (int n=0)
- XMLNode NewAttribute (const std::string &name)
- XMLNode NewAttribute (const char *name)
- XMLNode Attribute (const std::string &name)
- int AttributesSize (void)
- void Namespaces (const NS &namespaces)
- std::string NamespacePrefix (const char *urn)
- XMLNode NewChild (const std::string &name, int n=-1, bool global_order=false)

- XMLNode NewChild (const char *name, int n=-1, bool global_order=false)
- XMLNode NewChild (const XMLNode &node, int n=-1, bool global_order=false)
- void Destroy (void)

Protected Member Functions

- XMLNode (xmlNodePtr node)

Protected Attributes

- xmlNodePtr **node_**
- bool is_owner_
- bool is_temporary_

Friends

- bool MatchXMLName (const XMLNode &node1, const XMLNode &node2)
- bool MatchXMLName (const XMLNode &node, const char *name)

3.24.1 Detailed Description

Wrapper for LibXML library Tree interface. This class wraps XML Node, Document and Property/Attribute structures. Each instance serves as pointer to actual LibXML element and provides convenient (for chosen purpose) methods for manipulating it. This class has no special ties to LibXML library and may be easily rewritten for any XML parser which provides interface similar to LibXML Tree. It implements only small subset of XML capabilities, which is probably enough for performing most of useful actions. This class also filters out (usually) useless textual nodes which are often used to make XML documents human-readable.

3.24.2 Member Typedef Documentation

3.24.2.1 typedef std::map<std::string,std::string> Arc::XMLNode::NS

convenience typedef representing mapping between namespace URIs and their prefixes

3.24.3 Constructor & Destructor Documentation

3.24.3.1 Arc::XMLNode::XMLNode (xmlNodePtr *node*) [inline, protected]

Private constructor for inherited classes Creates instance and links to existing LibXML structure. Acquired structure is not owned by class instance. If there is need to completely pass control of LibXML document to then instance's is_owner_ variable has to be set to true.

3.24.3.2 Arc::XMLNode::XMLNode (void) [inline]

Constructor of invalid node Created instance does not point to XML element. All methods are still allowed for such instance but produce no results.

3.24.3.3 Arc::XMLNode::XMLNode (const XMLNode & *node*) [inline]

Copies existing instance. Underlying XML element is NOT copied. Ownership is NOT inherited.

3.24.3.4 Arc::XMLNode::XMLNode (const std::string & *xml*) [inline]

Creates XML document structure from textual representation of XML document. Created structure is pointed and owned by constructed instance

3.24.3.5 Arc::XMLNode::XMLNode (const char * *xml*, int *len* = -1) [inline]

Same as previous

3.24.3.6 Arc::XMLNode::XMLNode (const NS & *ns*) [inline]

Creates empty XML document structure with specified namespaces. Created structure is pointed and owned by constructed instance

3.24.3.7 Arc::XMLNode::~XMLNode (void) [inline]

Detructor Also destroys underlying XML document if owned by this instance

3.24.4 Member Function Documentation

3.24.4.1 XMLNode Arc::XMLNode::Attribute (const std::string & *name*)

Returns XMLNode instance representing first attribute of node with specified by name

3.24.4.2 XMLNode Arc::XMLNode::Attribute (int *n* = 0)

Returns XMLNode instance reresenting n-th attribute of node.

3.24.4.3 std::list<XMLNode> Arc::XMLNode::Attributes (void)

Returns list of all attributes of node

3.24.4.4 int Arc::XMLNode::AttributesSize (void)

Returns number of attributes of node

3.24.4.5 XMLNode Arc::XMLNode::Child (int *n* = 0) const [inline]

Returns XMLNode instance representing n-th child of XML element. If such does not exist invalid XMLNode instance is returned

3.24.4.6 void Arc::XMLNode::Destroy (void)

Destroys underlying XML element. XML element is unlinked from XML tree and destroyed. After this operation XMLNode instance becomes invalid

3.24.4.7 void Arc::XMLNode::GetXML (std::string & *xml*) const [inline]

Fills argument with this instance XML (sub)tree textual representation

Reimplemented in Arc::SOAPMessage.

3.24.4.8 void Arc::XMLNode::Name (std::string *name*) [inline]

Assign new name to XML node

3.24.4.9 std::string Arc::XMLNode::Name (void) const [inline]

Returns name of XML node

3.24.4.10 std::string Arc::XMLNode::NamespacePrefix (const char * *urn*)

Returns prefix of specified namespace. Empty string if no such namespace.

3.24.4.11 void Arc::XMLNode::Namespaces (const NS & *namespaces*)

Assign namespaces of XML document at point specified by this instance. If namespace already exists it gets new prefix. New namespaces are added. It is usefull to apply this method to XML being processed in order to refer to it's elements by known prefix.

Reimplemented in Arc::SOAPMessage.

3.24.4.12 XMLNode Arc::XMLNode::NewAttribute (const char * *name*)

Same as previous method

3.24.4.13 XMLNode Arc::XMLNode::NewAttribute (const std::string & *name*)

Creates new attribute with specified name.

3.24.4.14 XMLNode Arc::XMLNode::NewChild (const XMLNode & *node*, int *n* = -1, bool *global_order* = false)

Link a copy of supplied XML node as child. Returns instance refering to new child. XML element is a copy on supplied one but not ovned by returned instance

3.24.4.15 XMLNode Arc::XMLNode::NewChild (const char * *name*, int *n* = -1, bool *global_order* = false)

Same as previous method

3.24.4.16 XMLNode Arc::XMLNode::NewChild (const std::string & *name*, int *n* = -1, bool *global_order* = false) [inline]

Creates new child XML element at specified position with specified name. Default is to put it at end of list. If global order is true position applies to whole set of children, otherwise only to children of same name

3.24.4.17 Arc::XMLNode::operator bool (void) const [inline]

Returns true if instance points to XML element - valid instance

3.24.4.18 Arc::XMLNode::operator std::string (void) const [inline]

Returns textual content of node excluding content of children nodes

3.24.4.19 bool Arc::XMLNode::operator! (void) const [inline]

Returns true if instance does not point to XML element - invalid instance

3.24.4.20 XMLNode& Arc::XMLNode::operator= (const XMLNode & *node*) [inline]

Make instance refer to another XML node. Ownership is not inherited.

3.24.4.21 XMLNode& Arc::XMLNode::operator= (const char * *content*) [inline]

Same as previous method

3.24.4.22 XMLNode& Arc::XMLNode::operator= (const std::string & *content*) [inline]

Sets textual content of node. All existing children nodes are discarded.

3.24.4.23]

XMLNode Arc::XMLNode::operator[ ] (int *n*) const

Returns XMLNode instance representing n-th node in sequence of siblings of same name. It's main purpose is to be used to retrieve element in array of children of same name like node["name"][5]

3.24.4.24]

XMLNode Arc::XMLNode::operator[ ] (const std::string & *name*) const [inline]

Similar to previous method

3.24.4.25]

XMLNode Arc::XMLNode::operator[ ] (const char * *name*) const

Returns XMLNode instance representing first child element with specified name. Name may be "namespace_prefix:name" or simply "name". In last case namespace is ignored. If such node does not exist invalid XMLNode instance is returned

3.24.4.26 int Arc::XMLNode::Size (void) const [inline]

Returns number of children nodes

3.24.5 Friends And Related Function Documentation

3.24.5.1 bool MatchXMLName (const XMLNode & *node*, const char * *name*) [friend]

Returns true if 'name' matches name of 'node'. If name contains prefix it's checked too

3.24.5.2 bool MatchXMLName (const XMLNode & *node1*, const XMLNode & *node2*) [friend]

Returns true if underlying XML elements have same names

3.24.6 Member Data Documentation

3.24.6.1 bool Arc::XMLNode::is_owner_ [protected]

If true node is owned by this instance - hence released in destructor. Normally that may be true only for top level node of XML document.

3.24.6.2 bool Arc::XMLNode::is_temporary_ [protected]

This variable is for future

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

- XMLNode.h

Index