ARC::FlO Reference Manual

Generated by Doxygen 1.3.5

Wed Dec 7 09:40:23 2005

Contents

Chapter 1

ARC::FlO Class Index

1.1 ARC::FlO Class List

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

Chapter 2

ARC::FlO Class Documentation

2.1 FlOClient Class Reference

Interface to remote service.

`#include <client.h>`

Public Member Functions

- **FlOClient** (pFlOServiceRemote s)
- **FlOClient** (FlOServiceRemote &s)
- bool registr (FlOService &s)
- bool notify (std::list< FlOMessage > &msgs)

2.1.1 Detailed Description

Interface to remote service.

2.1.2 Member Function Documentation

2.1.2.1 bool FlOClient::notify (std::list< FlOMessage > & *msgs*)

Send notification to remote service.

2.1.2.2 bool FlOClient::registr (FlOService & *s*)

Register at remote service.

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

- client.h

2.2 FlOMessage Class Reference

Message sent between services for notification.

`#include <message.h>`

Public Member Functions

- **FlOMessage** (FlOObject::action_t t)
- **FlOMessage** (const FlOObject &o, FlOObject::action_t t)
- void serialize (std::string &msg)
- void operator= (FlOObject::action_t t)
- bool operator== (const std::string &id) const
- bool operator== (const FlOObject &o) const
- bool operator== (const_pFlOObject p) const
- FlOObject::action_t type (void) const
- const_pFlOObject object (void) const

Static Public Member Functions

- FlOMessage * deserialize (const std::string &msg)

2.2.1 Detailed Description

Message sent between services for notification.

2.2.2 Member Function Documentation

2.2.2.1 FlOMessage* FlOMessage::deserialize (const std::string & *msg*) [static]

Process received message. Created message have to be deleted.

2.2.2.2 const_pFlOObject FlOMessage::object (void) const [inline]

Get object associated with message.

2.2.2.3 void FlOMessage::operator= (FlOObject::action_t *t*) [inline]

Modify type of message.

2.2.2.4 bool FlOMessage::operator== (const_pFlOObject *p*) const [inline]

check if message refers to same object

2.2.2.5 bool FlOMessage::operator== (const FlOObject & *o*) const [inline]

Check if message refers to same object.

2.2.2.6 bool FlOMessage::operator== (const std::string & *id*) const [inline]

Check if message refers to object with same ID.

2.2.2.7 void FlOMessage::serialize (std::string & *msg*)

Convert message into form suitable for sending through channel.

2.2.2.8 FlOObject::action_t FlOMessage::type (void) const [inline]

Get type of message.

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

- message.h

2.3 FlOObject Class Reference

`#include <object.h>`

Public Types

- enum type_t { **undefined**, **reference**, **ordinary** }
- enum action_t { **unknown**, **created**, **updated**, **deleted** }

Public Member Functions

- FlOObject (const std::string &id, type_t t=undefined)
- FlOObject (FlOService &service, const std::string &id, type_t t=undefined)
- const std::string & id (void) const
- type_t type (void) const
- FlOService * service (void) const
- FlOService * service (FlOService *s)
- std::vector< FlOAttribute > & **attributes** (void)
- virtual operator bool (void)
- virtual bool operator! (void)
- virtual void serialize (XMLNode &x) const
- virtual void serialize (XMLNode &x, FlOMessage &m) const
- virtual void serialize (std::string &msg) const
- virtual void serialize (std::string &msg, FlOMessage &m) const
- virtual void notification (FlOMessage &msg, pFlOServiceRemote rsrv)
- virtual void maintain (void)
- virtual void notification (FlOService &srv, FlOMessage &msg, pFlOServiceRemote rsrv) const

Static Public Member Functions

- void **Register** (deserializer_t sr)
- FlOObject * deserialize (const XMLNode &x)
- FlOObject * deserialize (const XMLNode &x, FlOMessage &m)
- FlOObject * deserialize (const std::string &msg)
- FlOObject * deserialize (const std::string &msg, FlOMessage &m)

Protected Member Functions

- **FlOObject** (const FlOObject &o)

Protected Attributes

- std::vector< FlOAttribute > attrs_

Friends

- FlOObject * **deserialize_object** (const XMLNode &x)

2.3.1 Detailed Description

Class to represent both reference and ordinary objects with specific methods virtual. Real object reimplements one or another set of them.

2.3.2 Member Enumeration Documentation

2.3.2.1 enum FlOObject::action_t

State of object.

2.3.2.2 enum FlOObject::type_t

Kind of object.

2.3.3 Constructor & Destructor Documentation

2.3.3.1 FlOObject::FlOObject (const std::string & *id*, type_t *t* = undefined)

Create orphan object.

2.3.3.2 FlOObject::FlOObject (FlOService & *service*, const std::string & *id*, type_t *t* = undefined)

Create owned object.

2.3.4 Member Function Documentation

2.3.4.1 FlOObject* FlOObject::deserialize (const std::string & *msg*, FlOMessage & *m*) [static]

Process object recieved with message.

2.3.4.2 FlOObject* FlOObject::deserialize (const std::string & *msg*) [static]

Process receieved object.

2.3.4.3 FlOObject* FlOObject::deserialize (const XMLNode & *x*, FlOMessage & *m*) [static]

Convenience method to convert XML tree into object.

2.3.4.4 FlOObject* FlOObject::deserialize (const XMLNode & *x*) [static]

Convenience method to convert XML tree into object.

2.3.4.5 const std::string& FlOObject::id (void) const [inline]

Get unique id of object.

2.3.4.6 virtual void FlOObject::maintain (void) [virtual]

Called periodically by service to which this object belongs.

2.3.4.7 virtual void FlOObject::notification (FlOService & *srv*, FlOMessage & *msg*, pFlOServiceRemote *rsrv*) const [virtual]

Process received notification about unknown object (called method belongs to deserialized object).

2.3.4.8 virtual void FlOObject::notification (FlOMessage & *msg*, pFlOServiceRemote *rsrv*) [virtual]

Process received notification about this object.

2.3.4.9 virtual FlOObject::operator bool (void) [virtual]

If object is valid.

2.3.4.10 virtual bool FlOObject::operator! (void) [virtual]

If object is invalid.

2.3.4.11 virtual void FlOObject::serialize (std::string & *msg*, FlOMessage & *m*) const [virtual]

Serialize object for including into message.

2.3.4.12 virtual void FlOObject::serialize (std::string & *msg*) const [virtual]

Turn object into format suitable to send over wire.

2.3.4.13 virtual void FlOObject::serialize (XMLNode & *x*, FlOMessage & *m*) const [virtual]

Convenience method to convert object into XML tree Real object can reimplement it instead of one using string.

2.3.4.14 virtual void FlOObject::serialize (XMLNode & *x*) const [virtual]

Convenience method to convert object into XML tree. Real object can reimplement it instead of one using string.

2.3.4.15 FlOService* FlOObject::service (FlOService * *s*) [inline]

Assign to service (reassigning is not allowed).

2.3.4.16 FlOService* FlOObject::service (void) const [inline]

Get parent service.

2.3.4.17 type_t FlOObject::type (void) const [inline]

Get type of object.

2.3.5 Member Data Documentation

2.3.5.1 std::vector<FlOAttribute> FlOObject::attrs_ [protected]

Object attributes (general case).

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

- object.h

2.4 FlORegistration Class Reference

Registration request.

`#include <message.h>`

Public Member Functions

- **FlORegistration** (FlOService &s)
- FlOService & **service** (void)
- bool **operator==** (const FlOService &s) const

2.4.1 Detailed Description

Registration request.

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

- message.h

2.5 FlOService Class Reference

Kind of main class. Represents local service itself.

`#include <service.h>`

Public Member Functions

- **FlOService** (const std::string &url, const std::string &path)
- **operator bool** (void) const
- bool **operator!** (void) const
- const std::string & **url** (void) const
- void add (pFlOObject o)
- void add (pFlOServiceRemote s)
- void notify (const FlOMessage &m)
- void notification (FlOMessage &msg, pFlOServiceRemote &srv)
- pFlOServiceRemote service (const std::string &id)
- void registr (pFlOServiceRemote s)
- pFlOObject find (std::string id)
- void maintain (void)

2.5.1 Detailed Description

Kind of main class. Represents local service itself.

2.5.2 Member Function Documentation

2.5.2.1 void FlOService::add (pFlOServiceRemote *s*)

Add new service to be notified about new objects.

2.5.2.2 void FlOService::add (pFlOObject *o*)

Add new object to be handled.

2.5.2.3 pFlOObject FlOService::find (std::string *id*)

Find object by id (NULL if not found).

2.5.2.4 void FlOService::maintain (void)

Called to run internal tasks periodically.

2.5.2.5 void FlOService::notification (FlOMessage & *msg*, pFlOServiceRemote & *srv*)

Process recieved notification.

2.5.2.6 void FlOService::notify (const FlOMessage & *m*)

Inform registered services about changes (calls corresponding methods of hadled objects)

2.5.2.7 void FlOService::registr (pFlOServiceRemote *s*)

Register itself service to remote service.

2.5.2.8 pFlOServiceRemote FlOService::service (const std::string & *id*)

Find service in list by id or create a new one.

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

- service.h

2.6 FlOServiceRemote Class Reference

Interface to remote service.

`#include <service.h>`

Public Member Functions

- **FlOServiceRemote** (const std::string &url, const std::string &id)
- const std::string & **url** (void)
- const std::string & **id** (void)
- void notify (const FlOMessage &m)
- void registr (FlOService &s)
- void maintain (void)

2.6.1 Detailed Description

Interface to remote service.

2.6.2 Member Function Documentation

2.6.2.1 void FlOServiceRemote::maintain (void)

Called by parent service to run internal tasks periodically.

2.6.2.2 void FlOServiceRemote::notify (const FlOMessage & *m*)

Put messages into a queue (could also send it immediately).

2.6.2.3 void FlOServiceRemote::registr (FlOService & *s*)

Register s to this service.

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

- service.h

2.7 HTTP_FlO Class Reference

FlO service as seen by connected client (interface to FlOService class).

`#include <flo.h>`

Public Member Functions

- **HTTP_FlO** (HTTP_Connector *c, FlOService &srv)
- virtual HTTP_Error **get** (const char *uri, int &keep_alive)
- virtual HTTP_Error **put** (const char *uri, int &keep_alive)
- virtual void **soap_methods** (void)

Friends

- int nf__notification (struct soap *, int, char **, struct nf__response &)
- int nf__registration (struct soap *, char *, char *, char *, struct nf__response &)
- int **nf__add** (struct soap *, int, char **, struct nf__response &)
- int **nf__find** (struct soap *, char *, char *, struct nf__object_response &)

2.7.1 Detailed Description

FlO service as seen by connected client (interface to FlOService class).

2.7.2 Friends And Related Function Documentation

2.7.2.1 int nf__notification (struct soap *, int, char **, struct nf__response &) [friend]

Process notification from sent by other service.

2.7.2.2 int nf__registration (struct soap *, char *, char *, char *, struct nf__response &) [friend]

Process registration request from another service.

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

- flo.h

2.8 XMLNode Class Reference

`#include <xmlwrap.h>`

Public Member Functions

- XMLNode (void)
- XMLNode (const std::string &name)
- XMLNode (const XMLNode &node)
- XMLNode (XMLNode &parent, const std::string &name)
- operator bool (void) const
- bool operator! (void) const
- void operator= (const XMLNode &node)
- void addNode (const XMLNode &node)
- void linkNode (XMLNode &node)
- XMLNode * getNode (int n) const
- XMLNode * getNode (const std::string &name) const
- XMLNode * getNode (const std::string &name, int n) const
- int nodes (void) const
- std::string name (void) const
- void name (const std::string &s)
- std::string content (void) const
- void content (const std::string &s)
- std::string get (void) const
- void set (const std::string &s)

2.8.1 Detailed Description

Wrapper for sophisticated XML management API. Provides very simple, but also very simple to use functionality. There is no separate XML document. Everything is node. This implementation uses libxml2.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 XMLNode::XMLNode (void) `[inline]`

Empty tree.

2.8.2.2 XMLNode::XMLNode (const std::string & *name*)

Top named node.

2.8.2.3 XMLNode::XMLNode (const XMLNode & *node*)

Copy contructor (copies whole subtree).

2.8.2.4 XMLNode::XMLNode (XMLNode & *parent*, const std::string & *name*)

Named node will be attached to parent.

2.8.3 Member Function Documentation

2.8.3.1 void XMLNode::addNode (const XMLNode & *node*)

Attach copy of supplied node as a child.

2.8.3.2 void XMLNode::content (const std::string & *s*)

Set content of this node.

2.8.3.3 std::string XMLNode::content (void) const

Get content of node (inluding all children).

2.8.3.4 std::string XMLNode::get (void) const

Get XML subtree in printable format.

2.8.3.5 XMLNode* XMLNode::getNode (const std::string & *name*, int *n*) const

Get pointer to child node by name and order.

2.8.3.6 XMLNode* XMLNode::getNode (const std::string & *name*) const

Get pointer to child node by name (first found returned).

2.8.3.7 XMLNode* XMLNode::getNode (int *n*) const

Get pointer to child node by order number.

2.8.3.8 void XMLNode::linkNode (XMLNode & *node*)

Attach supplied node as a child (supplied node must be top node).

2.8.3.9 void XMLNode::name (const std::string & *s*)

Set name of this node.

2.8.3.10 std::string XMLNode::name (void) const

Get name of this node.

2.8.3.11 int XMLNode::nodes (void) const

Get number of child nodes.

2.8.3.12 XMLNode::operator bool (void) const [inline]

If node is not empty.

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

If node is empty.

2.8.3.14 void XMLNode::operator= (const XMLNode & *node*)

Replace this node with a copy of supplied one. This node becomes a top one.

2.8.3.15 void XMLNode::set (const std::string & *s*)

Set new XML tree. Node becomes top node.

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

- xmlwrap.h

Index