eProsima RPCDDS C++ API

Version 0.4.0

Generated by Doxygen 1.8.6

Mon Dec 1 2014 17:19:28

Contents

Chapter 1

eProsima RPC over DDS

eProsima RPCDDS Library

eProsima

eProsima RPC over DDS is a service invocation framework that enables to build distributed applications with minimal effort using the client/server paradigm. It makes transparent the remote procedure call to developer without the programmer explicitly coding the details for this remote interaction and allows developers to focus his efforts on their application logic.

eProsima RPC over DDS provides an easy way to invoke remote procedures using DDS standard as communication middleware. DDS (Data Distribution Service for Real-Time Systems) is an OMG specification of a data centric publish/subscribe communication model among real time software applications. eProsima RPC over DDS comes with all benefits that DDS standard provides as reliable and efficient communications for distributed real time systems.

eProsima RPC over DDS also brings other features:

- Synchronous, asynchronous and one-way invocations. The synchronous invocation is the common invocation and it blocks the client's thread until the reply is received from the server. The asynchronous invocation sends the request to the server but it doesn't blocks the client's thread. In the asynchronous invocation the developer provides a callback object that will be invoked when the reply is received from the server. The one-way invocation is a fire-and-forget invocation where the client does not care about the success or failure of the invocation. The one-way invocation does not expect any reply from the server.
- eProsima RPC over DDS provides several strategies for the server. These strategies define how the server acts when a new request is received. Current supported strategies are: single-thread strategy, thread-pool strategy and thread-per-request strategy. Single-thread strategy uses one thread for all incoming requests. Thread-pool strategy uses thread-pool's threads to process the incoming requests. Thread-per-request strategy creates a new thread for each new incoming request and this new thread will process the request.
- eProsima RPC over DDS supports several transports that DDS will use in the communications. There are two available transports. An UDP transport that brings the powerful benefit of DDS discovery in a local network or a TCP transport that allows connections with public servers located in internet.
- For DDS developers, eProsima RPC over DDS allows enhancing DDS with client/service communications. A developer that uses DDS in its distributed application will be able to use a service-oriented interaction too.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Module Documentation

5.1 eProsima RPCDDS API Reference

eProsima RPC over DDS internal API grouped in modules.

Collaboration diagram for eProsima RPCDDS API Reference:

Modules

- Client Module

  This group contains related API to create a client application. This API is used by the tool rpcddsgengen to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.
- Server Module

  This group contains related API to create a server application. Except the custom server's strategies, this API is used by the tool rpcddsgen to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.
- Exceptions

  Exceptions used by the eProsima RPCDDS API. All exceptions defined in this module are thrown by the eProsima RPCDDS library and the code generated by the tool rpcddsgen.
- Transports

  Network transports that eProsima RPCDDS library offers. These transports define how a connection is established between a proxy and a server.
- Protocols

  Protocols used by the RPCs. They define how to serialize and deserialize the information and use a eprosima::rpc::transport::Transport to send it and receive it.

5.1.1 Detailed Description

eProsima RPC over DDS internal API grouped in modules.

5.2 Client Module

This group contains related API to create a client application. This API is used by the tool *rpcddsgengen* to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.

Collaboration diagram for Client Module:

Classes

- class eprosima::rpc::proxy::Proxy

  This class implements the common functionalities that all server's proxies have.

5.2.1 Detailed Description

This group contains related API to create a client application. This API is used by the tool *rpcddsgengen* to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.

5.3 Server Module

This group contains related API to create a server application. Except the custom server's strategies, this API is used by the tool *rpcddsgen* to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.

Collaboration diagram for Server Module:

Modules

- Strategies

  Server's strategies that can be used in the server application. These strategies define how the server schedules a incoming request.

Classes

- class eprosima::rpc::server::Server

  This class implements the common functionalities that any server has.
- class eprosima::rpc::strategy::ServerStrategy

  This class is the base of all classes that implement a server strategy. that could be used by the server.
- class eprosima::rpc::strategy::ServerStrategyImpl

  This class is the base of all classes that implement a server strategy. that could be used by the server.

5.3.1 Detailed Description

This group contains related API to create a server application. Except the custom server's strategies, this API is used by the tool *rpcddsgen* to generate custom code based in a defined interface. That interface and its remote procedure have to be defined in IDL language.

5.4 Exceptions

Exceptions used by the eProsima RPCDDS API. All exceptions defined in this module are thrown by the eProsima RPCDDS library and the code generated by the tool *rpcddsgen*.

Collaboration diagram for Exceptions:

Classes

- class eprosima::rpc::exception::BadParamException

  This class is thrown as an exception when there is some bad paremeter in a object.
- class eprosima::rpc::exception::ClientInternalException

  This class is thrown as an exception when there is an error in the proxy side.
- class eprosima::rpc::exception::Exception

  This abstract class is used to create exceptions.
- class eprosima::rpc::exception::IncompatibleException

  This class is thrown as an exception when a selected protocol and transport are incompatible.
- class eprosima::rpc::exception::InitializeException

  This class is thrown as an exception when there is an error initializating an object.
- class eprosima::rpc::exception::ServerInternalException

  This class is thrown as an exception when there is an error in the server side.
- class eprosima::rpc::exception::ServerNotFoundException

  This class is thrown as an exception when the server is not found.
- class eprosima::rpc::exception::ServerTimeoutException

  This class is thrown as an exception when the remote procedure call exceeds the maximum time.
- class eprosima::rpc::exception::SystemException

  This abstract class is used to create internal FASTRPC exceptions.
- class eprosima::rpc::exception::UserException

  This abstract class is used to create user exceptions.

5.4.1 Detailed Description

Exceptions used by the eProsima RPCDDS API. All exceptions defined in this module are thrown by the eProsima RPCDDS library and the code generated by the tool *rpcddsgen*.

5.5 Strategies

Server's strategies that can be used in the server application. These strategies define how the server schedules a incoming request.

Collaboration diagram for Strategies:

Classes

- class eprosima::rpc::strategy::SingleThreadStrategy

  This class implements the sigle thread strategy. The server uses a reception thread to execute all the requests.
- class eprosima::rpc::strategy::ThreadPerRequestStrategy

  This class implements the thread per request strategy. The server creates a new thread for every new incoming request.
- class eprosima::rpc::strategy::ThreadPoolStrategy

  This class implements a thread pool strategy. The server schedules the incoming requests in a free thread of the thread pool.

5.5.1 Detailed Description

Server's strategies that can be used in the server application. These strategies define how the server schedules a incoming request.

5.6 Transports

Network transports that eProsima RPCDDS library offers. These transports define how a connection is established between a proxy and a server.

Collaboration diagram for Transports:

Classes

- class eprosima::rpc::transport::AsyncTask

  This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.
- class eprosima::rpc::transport::Endpoint

  This class represents an endpoint.
- class eprosima::rpc::transport::dds::AsyncThread

  This class is a separated thread used to manage asynchronous tasks.
- class eprosima::rpc::transport::dds::ProxyProcedureEndpoint

  This class represents a remote endpoint used by a proxy. It also encapsulates the DDS datawriter and the DDS datareader.
- class eprosima::rpc::transport::dds::ServerProcedureEndpoint

  This class represents a remote endpoint used by a proxy. Also this class encapsulate the DDS datawriter and the DDS datareader.
- class eprosima::rpc::transport::dds::DDSAsyncTask

  This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.
- class eprosima::rpc::transport::dds::ProxyTransport

  This class is the base of all proxies that implement a transport using DDS.
- class eprosima::rpc::transport::dds::ServerTransport

  This class is the base of all classes that implement a transport using DDS. This transport can be used by the servers.
- class eprosima::rpc::transport::dds::TCPProxyTransport

  This class implements a transport using DDS over TCPv4. This transport can only be used by a server proxy.
- class eprosima::rpc::transport::dds::TCPServerTransport

  This class implements a transport using DDS over TCPv4. This transport can only be used by a server.
- class eprosima::rpc::transport::dds::Transport

  This class is the base of all classes that implement a transport using DDS. This transport could be used by both proxies and servers.
- class eprosima::rpc::transport::dds::UDPProxyTransport

  This class implements a transport using DDS over UDPv4. This transport only can be used by a server's proxy.
- class eprosima::rpc::transport::dds::UDPServerTransport

  This class implements transport using DDS over UDPv4. This transport can only be used by a server.

Typedefs

- typedef enum eprosima::rpc::transport::TransportBehaviour eprosima::rpc::transport::TransportBehaviour

  This enumeration specifies the behaviour of the transport.

Enumerations

- enum eprosima::rpc::transport::TransportBehaviour { **PROXY_BEHAVIOUR**, **SERVER_BEHAVIOUR** }

  This enumeration specifies the behaviour of the transport.

5.6.1 Detailed Description

Network transports that eProsima RPCDDS library offers. These transports define how a connection is established between a proxy and a server.

5.7 Protocols

Protocols used by the RPCs. They define how to serialize and deserialize the information and use a eprosima::rpc::transport::Transport to send it and receive it.

Collaboration diagram for Protocols:

Classes

- class eprosima::rpc::protocol::dds::SampleIdentity_t

  This class is used to identify clients.
- class eprosima::rpc::protocol::dds::RequestHeader

  Header information used in all generated request topics.
- class eprosima::rpc::protocol::dds::ReplyHeader

  Header information used in all generated reply topics.
- class eprosima::rpc::protocol::dds::RequestHeaderPlugin

  This class offers the functions needed by DDS middleware to use the class RequestHeaderPlugin.
- class eprosima::rpc::protocol::dds::ReplyHeaderPlugin

  This class offers the functions needed by DDS middleware to use the class ReplyHeaderPlugin.
- class eprosima::rpc::protocol::Protocol

  This abstract class represents the protocol used by the RPCs. It serializes and deserializes the information and uses a eprosima::rpc::transport::Transport to send it and receive it.

5.7.1 Detailed Description

Protocols used by the RPCs. They define how to serialize and deserialize the information and use a eprosima::rpc::transport::Transport to send it and receive it.

5.8 Generated API example for eProsima RPCDDS

This group contains the generated API by the tool *rpcddsgen for* a DDS example of an interface named Foo.

Classes

- interface FooDDS::Foo

  This class represents the interface Foo defined by the user in the IDL file.
- class FooDDS::Foo_FooProcedureCallbackHandler

  This abstract class defines the callbacks that eProsima RPC will call in an asynchronous call. These callback has to be implemented in a derived class.
- class FooDDS::Foo_FooProcedureTask

  This class represents a asynchronous task created to wait the reply of the procedure Foo::FooProcedure from the server in an asynchronous call.
- class eprosima::rpc::protocol::dds::FooDDSProtocol

  This class is responsible for serializing and deserializing the requests and responses of this application. It uses DDS.
- class FooDDS::FooProxy

  This class implements a specific server's proxy for the defined interface Foo.
- class FooDDS::FooServer

  This class implements a specific server for the defined interface Foo by user.
- class FooDDS::FooServerImpl

  This class is the skeleton of the servant and its remote procedures has to be implemented.
- class FooDDS::Foo_FooProcedure_In

  This class represents the structure Foo_FooProcedure_In that can be used to send/receive requests for the operation Foo::FooProcedure.
- class FooDDS::Foo_FooProcedure_Out

  This class encapsulates output paramaters for operation Foo::FooProcedure.
- class FooDDS::Foo_FooProcedure_Result

  This class represents the structure Foo_FooProcedure_Result that can be used to send/receive replies for the operation Foo::FooProcedure.
- class FooDDS::Foo_Call

  This class represents the union used in the DDS topic to encapsulate the operations in request samples.
- class FooDDS::Foo_Request

  This class represents the structure Foo_Request that can be used to send/receive requests for the interface Foo.
- class FooDDS::Foo_Return

  This class represents the union used in the DDS topic to encapsulate the operations in reply samples.
- class FooDDS::Foo_Reply

  This class represents the structure Foo_Reply that can be used to send/receive replies for the interface Foo.

5.8.1 Detailed Description

This group contains the generated API by the tool *rpcddsgen for* a DDS example of an interface named Foo.

Chapter 6

Class Documentation

6.1 eprosima::rpc::transport::AsyncTask Class Reference

This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.

`#include <AsyncTask.h>`

Inheritance diagram for eprosima::rpc::transport::AsyncTask:

Protected Member Functions

- AsyncTask ()

  Default constructor.
- virtual ~AsyncTask ()

  Destructor.

6.1.1 Detailed Description

This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.

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

- includetmp/rpcdds/transports/AsyncTask.h

6.2 eprosima::rpc::transport::dds::AsyncThread Class Reference

This class is a separated thread used to manage asynchronous tasks.

`#include <AsyncThread.h>`

Public Member Functions

- AsyncThread ()

  Default constructor.
- int init ()

  This function initializes all internal objects.
- void exit ()

  This function deletes the internal objects.

- int addTask (DDSQueryCondition *query, DDSAsyncTask *task, long timeout)

  This function adds a new asynchronous task.
- void deleteAssociatedAsyncTasks (ProxyProcedureEndpoint *pe)

  This function deletes all the asynchronous tasks associated with the ProxyProcedureEndpoint endpoint.

6.2.1 Detailed Description

This class is a separated thread used to manage asynchronous tasks.

6.2.2 Member Function Documentation

6.2.2.1 int eprosima::rpc::transport::dds::AsyncThread::addTask (DDSQueryCondition * *query*, DDSAsyncTask * *task*, long *timeout*)

This function adds a new asynchronous task.

Parameters

query	Associated DDS::QueryCondition to the asynchronous task. Cannot be NULL.
task	The new asynchronous task. Cannot be NULL.
timeout	The time in milliseconds to wait for the reply.

Returns

0 if the function succesfully works. -1 in other case

6.2.2.2 void eprosima::rpc::transport::dds::AsyncThread::deleteAssociatedAsyncTasks (ProxyProcedureEndpoint * *pe*)

This function deletes all the asynchronous tasks associated with the ProxyProcedureEndpoint endpoint.

Parameters

pe	Pointer to the ProxyProcedureEndpoint. It cannot be NULL.

6.2.2.3 int eprosima::rpc::transport::dds::AsyncThread::init ( )

This function initializes all internal objects.

Returns

0 value is returned if all the objects was succesfully created. -1 in other case

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

- includetmp/rpcdds/transports/dds/AsyncThread.h

6.3 eprosima::rpc::exception::BadParamException Class Reference

This class is thrown as an exception when there is some bad paremeter in a object.

`#include <BadParamException.h>`

Inheritance diagram for eprosima::rpc::exception::BadParamException:

Collaboration diagram for eprosima::rpc::exception::BadParamException:

Public Member Functions

- FASTRPC_DllAPI BadParamException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI BadParamException (const BadParamException &ex)

  Default copy constructor.
- FASTRPC_DllAPI BadParamException (BadParamException &&ex)

  Default move constructor.
- FASTRPC_DllAPI BadParamException & operator= (const BadParamException &ex)

  Assigment operation.
- FASTRPC_DllAPI BadParamException & operator= (BadParamException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~BadParamException () throw ()

  Default constructor.
- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.3.1 Detailed Description

This class is thrown as an exception when there is some bad paremeter in a object.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 FASTRPC_DllAPI eprosima::rpc::exception::BadParamException::BadParamException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.3.2.2 FASTRPC_DllAPI eprosima::rpc::exception::BadParamException::BadParamException (const BadParamException & *ex*)

Default copy constructor.

Parameters

ex	BadParamException that will be copied.

6.3.2.3 FASTRPC_DllAPI eprosima::rpc::exception::BadParamException::BadParamException (BadParamException && *ex*)

Default move constructor.

Parameters

ex	BadParamException that will be moved.

6.3.3 Member Function Documentation

6.3.3.1 FASTRPC_DllAPI BadParamException& eprosima::rpc::exception::BadParamException::operator= (const BadParamException & *ex*)

Assigment operation.

Parameters

ex	BadParamException that will be copied.

6.3.3.2 FASTRPC_DllAPI BadParamException& eprosima::rpc::exception::BadParamException::operator= (BadParamException && *ex*)

Assigment operation.

Parameters

ex	BadParamException that will be moved.

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

- includetmp/rpcdds/exceptions/BadParamException.h

6.4 eprosima::rpc::exception::ClientInternalException Class Reference

This class is thrown as an exception when there is an error in the proxy side.

`#include <ClientInternalException.h>`

Inheritance diagram for eprosima::rpc::exception::ClientInternalException:

Collaboration diagram for eprosima::rpc::exception::ClientInternalException:

Public Member Functions

- FASTRPC_DllAPI ClientInternalException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI ClientInternalException (const ClientInternalException &ex)

  Default copy constructor.
- FASTRPC_DllAPI ClientInternalException (ClientInternalException &&ex)

  Default move constructor.
- FASTRPC_DllAPI ClientInternalException & operator= (const ClientInternalException &ex)

  Assigment operation.
- FASTRPC_DllAPI ClientInternalException & operator= (ClientInternalException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~ClientInternalException () throw ()

  Default constructor.
- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.4.1 Detailed Description

This class is thrown as an exception when there is an error in the proxy side.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 FASTRPC_DllAPI eprosima::rpc::exception::ClientInternalException::ClientInternalException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.4.2.2 FASTRPC_DllAPI eprosima::rpc::exception::ClientInternalException::ClientInternalException (const ClientInternalException & *ex*)

Default copy constructor.

Parameters

ex	ClientInternalException that will be copied.

6.4.2.3 FASTRPC_DllAPI eprosima::rpc::exception::ClientInternalException::ClientInternalException (ClientInternalException && *ex*)

Default move constructor.

Parameters

ex	ClientInternalException that will be moved.

6.4.3 Member Function Documentation

6.4.3.1 FASTRPC_DllAPI ClientInternalException& eprosima::rpc::exception::ClientInternalException::operator= (const ClientInternalException & *ex*)

Assigment operation.

Parameters

ex	ClientInternalException that will be copied.

6.4.3.2 FASTRPC_DllAPI ClientInternalException& eprosima::rpc::exception::ClientInternalException::operator= (ClientInternalException && *ex*)

Assigment operation.

Parameters

ex	ClientInternalException that will be moved.

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

- includetmp/rpcdds/exceptions/ClientInternalException.h

6.5 eprosima::rpc::transport::dds::DDSAsyncTask Class Reference

This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.

`#include <DDSAsyncTask.h>`

Inheritance diagram for eprosima::rpc::transport::dds::DDSAsyncTask:

Collaboration diagram for eprosima::rpc::transport::dds::DDSAsyncTask:

Public Member Functions

- DDSAsyncTask ()

  default constructor
- virtual ~DDSAsyncTask ()

  default destructor
- void execute (DDSQueryCondition *query)

  This function is called when the DDS WaitSet was wake up by the query condition of this asynchronous task. This funtion takes the reply.
- void setProcedureEndpoint (ProxyProcedureEndpoint *pe)

  Sets the procedure endpoint.
- ProxyProcedureEndpoint * getProcedureEndpoint ()

  Gets the procedure endpoint.
- virtual void on_exception (const exception::SystemException &ex)=0

  This function executes the callback function when an exception occurs on the client's side. This function should be implemented by the generated asynchronous tasks.

Protected Member Functions

- virtual void execute ()=0

  This function executes the callback functions when a reply is received or an exception was transmitted. This function should be implemented by the generated asynchronous tasks.
- virtual void * getReplyInstance ()=0

  Returns the allocated memory that will be used when the reply is taken.

6.5.1 Detailed Description

This class represents a asynchronous task created to wait the reply from the server in an asynchronous call.

6.5.2 Member Function Documentation

6.5.2.1 void eprosima::rpc::transport::dds::DDSAsyncTask::execute (DDSQueryCondition * *query*)

This function is called when the DDS WaitSet was wake up by the query condition of this asynchronous task. This funtion takes the reply.

Parameters

query	Query condition associated with this asynchronous task.

6.5.2.2 ProxyProcedureEndpoint* eprosima::rpc::transport::dds::DDSAsyncTask::getProcedureEndpoint ( )

Gets the procedure endpoint.

Returns

Procedure endpoint with the DDS datawriter and datareader

6.5.2.3 virtual void* eprosima::rpc::transport::dds::DDSAsyncTask::getReplyInstance ( ) [protected], [pure virtual]

Returns the allocated memory that will be used when the reply is taken.

Returns

Pointer to the allocated memory.

Implemented in FooDDS::Foo_FooProcedureTask.

6.5.2.4 virtual void eprosima::rpc::transport::dds::DDSAsyncTask::on_exception (const exception::SystemException & *ex*) [pure virtual]

This function executes the callback function when an exception occurs on the client's side. This function should be implemented by the generated asynchronous tasks.

Parameters

ex	The exception that is sent to the user.

Implemented in FooDDS::Foo_FooProcedureTask.

6.5.2.5 void eprosima::rpc::transport::dds::DDSAsyncTask::setProcedureEndpoint (ProxyProcedureEndpoint * *pe*)

Sets the procedure endpoint.

Parameters

pe	Procedure endpoint with the DDS datawriter and datareader

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

- includetmp/rpcdds/transports/dds/DDSAsyncTask.h

6.6 eprosima::rpc::transport::Endpoint Class Reference

This class represents an endpoint.

`#include <Endpoint.h>`

Inheritance diagram for eprosima::rpc::transport::Endpoint:

Protected Member Functions

- Endpoint ()

  Default constructor.
- virtual ~Endpoint ()

  Default destructor.

6.6.1 Detailed Description

This class represents an endpoint.

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

- includetmp/rpcdds/transports/components/Endpoint.h

6.7 eprosima::rpc::exception::Exception Class Reference

This abstract class is used to create exceptions.

`#include <Exception.h>`

Inheritance diagram for eprosima::rpc::exception::Exception:

Collaboration diagram for eprosima::rpc::exception::Exception:

Public Member Functions

- virtual ~Exception () throw ()

  Default destructor.
- virtual void raise () const =0

  This function throws the object as exception.

Protected Member Functions

- Exception ()

  Default constructor.
- Exception (const Exception &ex)

  Default copy constructor.
- Exception (Exception &&ex)

  Default move constructor.
- Exception & operator= (const Exception &ex)

  Assigment operation.
- Exception & operator= (Exception &&)

  Assigment operation.

6.7.1 Detailed Description

This abstract class is used to create exceptions.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 eprosima::rpc::exception::Exception::Exception (const Exception & *ex*) [protected]

Default copy constructor.

Parameters

ex	Exception that will be copied.

6.7.2.2 eprosima::rpc::exception::Exception::Exception (Exception && *ex*) [protected]

Default move constructor.

Parameters

ex	Exception that will be moved.

6.7.3 Member Function Documentation

6.7.3.1 Exception& eprosima::rpc::exception::Exception::operator= (const Exception & *ex*) [protected]

Assigment operation.

Parameters

ex	Exception that will be copied.

6.7.3.2 Exception& eprosima::rpc::exception::Exception::operator= (Exception &&) [protected]

Assigment operation.

Parameters

ex	Exception that will be moved.

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

- includetmp/rpcdds/exceptions/Exception.h

6.8 FooDDS::Foo Interface Reference

This class represents the interface Foo defined by the user in the IDL file.

`#include <FooDDS.h>`

Inheritance diagram for FooDDS::Foo:

Public Member Functions

- virtual void **FooProcedure** ()=0
- void **FooProcedure** ()

6.8.1 Detailed Description

This class represents the interface Foo defined by the user in the IDL file.

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

- utils/doxygen/examples/dds/FooDDS.h
- utils/doxygen/examples/dds/FooDDS.idl

6.9 FooDDS::Foo_Call Class Reference

This class represents the union used in the DDS topic to encapsulate the operations in request samples.

`#include <FooDDSTopics.h>`

Public Member Functions

- user_cpp_DllExport Foo_Call ()
  Default constructor.
- user_cpp_DllExport ~Foo_Call ()
  Destructor.
- user_cpp_DllExport Foo_Call (const Foo_Call &x)
  Copy constructor.
- user_cpp_DllExport Foo_Call (Foo_Call &&x)
  Move constructor.
- user_cpp_DllExport Foo_Call & operator= (const Foo_Call &x)
  Copy assignment.
- user_cpp_DllExport Foo_Call & operator= (Foo_Call &&x)
  Move assignment.
- user_cpp_DllExport void _d (int32_t __d)
  This function sets the discriminator value.
- user_cpp_DllExport int32_t _d () const
  This function returns the value of the discriminator.
- user_cpp_DllExport int32_t & _d ()
  This function returns a reference to the discriminator.
- user_cpp_DllExport void **unknown_operation** (eprosima::rpc::protocol::dds::UnknownOperation _unknown_operation)
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownOperation **unknown_operation** () const
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownOperation & **unknown_operation** ()
- user_cpp_DllExport void FooProcedure (const Foo_FooProcedure_In &_FooProcedure)
  This function copies the value in member FooProcedure.
- user_cpp_DllExport void FooProcedure (Foo_FooProcedure_In &&_FooProcedure)
  This function moves the value in member FooProcedure.
- user_cpp_DllExport const Foo_FooProcedure_In & FooProcedure () const
  This function returns a constant reference to member FooProcedure.
- user_cpp_DllExport Foo_FooProcedure_In & FooProcedure ()
  This function returns a reference to member FooProcedure.
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const
  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const
  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)
  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)
  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.9.1 Detailed Description

This class represents the union used in the DDS topic to encapsulate the operations in request samples.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 FooDDS::Foo_Call::Foo_Call (const Foo_Call & *x*)

Copy constructor.

Parameters

x	Reference to the object Foo_Call that will be copied.

6.9.2.2 FooDDS::Foo_Call::Foo_Call (Foo_Call && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_Call that will be copied.

6.9.3 Member Function Documentation

6.9.3.1 void FooDDS::Foo_Call::_d (int32_t *__d*)

This function sets the discriminator value.

Parameters

__d	New value for the discriminator.

Exceptions

eprosima::rpc::exception::BadParamException	This exception is thrown if the new value doesn't correspond to the selected union member.

6.9.3.2 int32_t FooDDS::Foo_Call::_d ( ) const

This function returns the value of the discriminator.

Returns

Value of the discriminator

6.9.3.3 int32_t & FooDDS::Foo_Call::_d ( )

This function returns a reference to the discriminator.

Returns

Reference to the discriminator.

6.9.3.4 void FooDDS::Foo_Call::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.9.3.5 void FooDDS::Foo_Call::FooProcedure (const Foo_FooProcedure_In & *_FooProcedure*)

This function copies the value in member FooProcedure.

Parameters

_FooProcedure	New value to be copied in member FooProcedure

6.9.3.6 void FooDDS::Foo_Call::FooProcedure (FooDDS::Foo_FooProcedure_In && *_FooProcedure*)

This function moves the value in member FooProcedure.

Parameters

_FooProcedure	New value to be moved in member FooProcedure

6.9.3.7 const FooDDS::Foo_FooProcedure_In & FooDDS::Foo_Call::FooProcedure ( ) const

This function returns a constant reference to member FooProcedure.

Returns

Constant reference to member FooProcedure

6.9.3.8 FooDDS::Foo_FooProcedure_In & FooDDS::Foo_Call::FooProcedure ( )

This function returns a reference to member FooProcedure.

Returns

Reference to member FooProcedure

6.9.3.9 size_t FooDDS::Foo_Call::getMaxCdrSerializedSize (size_t *current_alignment* = `0`) `[static]`

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.9.3.10 size_t FooDDS::Foo_Call::getSerializedSize (size_t *current_alignment* = `0`) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.9.3.11 FooDDS::Foo_Call & FooDDS::Foo_Call::operator= (const Foo_Call & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_Call that will be copied.

6.9.3.12 FooDDS::Foo_Call & FooDDS::Foo_Call::operator= (Foo_Call && *x*)

Move assignment.

Parameters

x	Reference to the object Foo_Call that will be copied.

6.9.3.13 void FooDDS::Foo_Call::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.10 FooDDS::Foo_CallPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * **get_typecode** ()

6.10.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.11 FooDDS::Foo_FooProcedure_In Class Reference

This class represents the structure Foo_FooProcedure_In that can be used to send/receive requests for the operation Foo::FooProcedure.

`#include <FooDDSTopics.h>`

Public Member Functions

- user_cpp_DllExport Foo_FooProcedure_In ()

  Default constructor.
- user_cpp_DllExport ~Foo_FooProcedure_In ()

  Destructor.
- user_cpp_DllExport **Foo_FooProcedure_In** (const Foo_FooProcedure_In &x)
- user_cpp_DllExport Foo_FooProcedure_In (Foo_FooProcedure_In &&x)

  Move constructor.
- user_cpp_DllExport Foo_FooProcedure_In & operator= (const Foo_FooProcedure_In &x)

  Copy assignment.
- user_cpp_DllExport Foo_FooProcedure_In & operator= (Foo_FooProcedure_In &&x)

  Copy assignment.
- user_cpp_DllExport void **dummy** (int32_t _dummy)
- user_cpp_DllExport int32_t **dummy** () const
- user_cpp_DllExport int32_t & **dummy** ()
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const

  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)

  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.11.1 Detailed Description

This class represents the structure Foo_FooProcedure_In that can be used to send/receive requests for the operation Foo::FooProcedure.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 FooDDS::Foo_FooProcedure_In::Foo_FooProcedure_In (Foo_FooProcedure_In && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_FooProcedure_In that will be copied.

6.11.3 Member Function Documentation

6.11.3.1 void FooDDS::Foo_FooProcedure_In::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.11.3.2 size_t FooDDS::Foo_FooProcedure_In::getMaxCdrSerializedSize (size_t *current_alignment* = 0) [static]

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.11.3.3 size_t FooDDS::Foo_FooProcedure_In::getSerializedSize (size_t *current_alignment* = 0) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.11.3.4 FooDDS::Foo_FooProcedure_In & FooDDS::Foo_FooProcedure_In::operator= (const Foo_FooProcedure_In & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.11.3.5 FooDDS::Foo_FooProcedure_In & FooDDS::Foo_FooProcedure_In::operator= (FooDDS::Foo_FooProcedure_In && *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.11.3.6 void FooDDS::Foo_FooProcedure_In::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.12 FooDDS::Foo_FooProcedure_Out Class Reference

This class encapsulates output paramaters for operation Foo::FooProcedure.

`#include <FooDDSTopics.h>`

Public Member Functions

- user_cpp_DllExport Foo_FooProcedure_Out ()
  Default constructor.
- user_cpp_DllExport ~Foo_FooProcedure_Out ()
  Destructor.
- user_cpp_DllExport Foo_FooProcedure_Out (const Foo_FooProcedure_Out &x)
  Copy constructor.
- user_cpp_DllExport Foo_FooProcedure_Out (Foo_FooProcedure_Out &&x)
  Move constructor.
- user_cpp_DllExport Foo_FooProcedure_Out & operator= (const Foo_FooProcedure_Out &x)
  Copy assignment.
- user_cpp_DllExport Foo_FooProcedure_Out & operator= (Foo_FooProcedure_Out &&x)
  Copy assignment.
- user_cpp_DllExport void **dummy** (int32_t _dummy)
- user_cpp_DllExport int32_t **dummy** () const
- user_cpp_DllExport int32_t & **dummy** ()
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const
  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const
  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)
  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)
  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.12.1 Detailed Description

This class encapsulates output paramaters for operation Foo::FooProcedure.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 FooDDS::Foo_FooProcedure_Out::Foo_FooProcedure_Out (const Foo_FooProcedure_Out & *x*)

Copy constructor.

Parameters

x	Reference to the object Foo_FooProcedure_Out that will be copied.

6.12.2.2 FooDDS::Foo_FooProcedure_Out::Foo_FooProcedure_Out (Foo_FooProcedure_Out && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_FooProcedure_Out that will be copied.

6.12.3 Member Function Documentation

6.12.3.1 void FooDDS::Foo_FooProcedure_Out::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.12.3.2 size_t FooDDS::Foo_FooProcedure_Out::getMaxCdrSerializedSize (size_t *current_alignment* = 0) [static]

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.12.3.3 size_t FooDDS::Foo_FooProcedure_Out::getSerializedSize (size_t *current_alignment* = 0) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.12.3.4 FooDDS::Foo_FooProcedure_Out & FooDDS::Foo_FooProcedure_Out::operator= (const Foo_FooProcedure_Out & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.12.3.5 FooDDS::Foo_FooProcedure_Out & FooDDS::Foo_FooProcedure_Out::operator= (FooDDS::Foo_FooProcedure_Out && *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.12.3.6 void FooDDS::Foo_FooProcedure_Out::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.13 FooDDS::Foo_FooProcedure_Result Class Reference

This class represents the structure Foo_FooProcedure_Result that can be used to send/receive replies for the operation Foo::FooProcedure.

```
#include <FooDDSTopics.h>
```

Public Member Functions

- user_cpp_DllExport Foo_FooProcedure_Result ()
  Default constructor.
- user_cpp_DllExport ~Foo_FooProcedure_Result ()
  Destructor.
- user_cpp_DllExport Foo_FooProcedure_Result (const Foo_FooProcedure_Result &x)
  Copy constructor.
- user_cpp_DllExport Foo_FooProcedure_Result (Foo_FooProcedure_Result &&x)
  Move constructor.
- user_cpp_DllExport Foo_FooProcedure_Result & operator= (const Foo_FooProcedure_Result &x)

Copy assignment.

- user_cpp_DllExport Foo_FooProcedure_Result & operator= (Foo_FooProcedure_Result &&x)

  Copy assignment.
- user_cpp_DllExport void _d (int32_t __d)

  This function sets the discriminator value.
- user_cpp_DllExport int32_t _d () const

  This function returns the value of the discriminator.
- user_cpp_DllExport int32_t & _d ()

  This function returns a reference to the discriminator.
- user_cpp_DllExport void **unknown_exception** (eprosima::rpc::protocol::dds::UnknownException _unknown_exception)
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownException **unknown_exception** () const
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownException & **unknown_exception** ()
- user_cpp_DllExport void out_ (const Foo_FooProcedure_Out &_out_)

  This function copies the value in member out_.
- user_cpp_DllExport void out_ (Foo_FooProcedure_Out &&_out_)

  This function moves the value in member out_.
- user_cpp_DllExport const Foo_FooProcedure_Out & out_ () const

  This function returns a constant reference to member out_.
- user_cpp_DllExport Foo_FooProcedure_Out & out_ ()

  This function returns a reference to member out_.
- user_cpp_DllExport void **sysx_** (eprosima::rpc::ReturnMessage _sysx_)
- user_cpp_DllExport eprosima::rpc::ReturnMessage **sysx_** () const
- user_cpp_DllExport eprosima::rpc::ReturnMessage & **sysx_** ()
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const

  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)

  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.13.1 Detailed Description

This class represents the structure Foo_FooProcedure_Result that can be used to send/receive replies for the operation Foo::FooProcedure.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 FooDDS::Foo_FooProcedure_Result::Foo_FooProcedure_Result (const Foo_FooProcedure_Result & *x*)

Copy constructor.

Parameters

x	Reference to the object Foo_FooProcedure_Result that will be copied.

6.13.2.2 FooDDS::Foo_FooProcedure_Result::Foo_FooProcedure_Result (Foo_FooProcedure_Result && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_FooProcedure_Result that will be copied.

6.13.3 Member Function Documentation

6.13.3.1 void FooDDS::Foo_FooProcedure_Result::_d (int32_t *__d*)

This function sets the discriminator value.

Parameters

__d	New value for the discriminator.

Exceptions

eprosima::rpc::exception::BadParamException	This exception is thrown if the new value doesn't correspond to the selected union member.

6.13.3.2 int32_t FooDDS::Foo_FooProcedure_Result::_d ( ) const

This function returns the value of the discriminator.

Returns

Value of the discriminator

6.13.3.3 int32_t & FooDDS::Foo_FooProcedure_Result::_d ( )

This function returns a reference to the discriminator.

Returns

Reference to the discriminator.

6.13.3.4 void FooDDS::Foo_FooProcedure_Result::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.13.3.5 size_t FooDDS::Foo_FooProcedure_Result::getMaxCdrSerializedSize (size_t *current_alignment* = 0) [static]

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.13.3.6 size_t FooDDS::Foo_FooProcedure_Result::getSerializedSize (size_t *current_alignment* = 0) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.13.3.7 FooDDS::Foo_FooProcedure_Result & FooDDS::Foo_FooProcedure_Result::operator= (const Foo_FooProcedure_Result & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.13.3.8 FooDDS::Foo_FooProcedure_Result & FooDDS::Foo_FooProcedure_Result::operator= (FooDDS::Foo_FooProcedure_Result && *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_FooProcedure that will be copied.

6.13.3.9 void FooDDS::Foo_FooProcedure_Result::out_ (const Foo_FooProcedure_Out & *_out_*)

This function copies the value in member out_.

Parameters

out	New value to be copied in member out_

6.13.3.10 void FooDDS::Foo_FooProcedure_Result::out_ (FooDDS::Foo_FooProcedure_Out && *_out_*)

This function moves the value in member out_.

Parameters

out	New value to be moved in member out_

6.13.3.11 const FooDDS::Foo_FooProcedure_Out & FooDDS::Foo_FooProcedure_Result::out_ ( ) const

This function returns a constant reference to member out_.

Returns

Constant reference to member out_

Exceptions

eprosima::rpc::exception::BadParamException	This exception is thrown if the requested union member is not the current selection.

6.13.3.12 FooDDS::Foo_FooProcedure_Out & FooDDS::Foo_FooProcedure_Result::out_ ( )

This function returns a reference to member out_.

Returns

Reference to member out_

Exceptions

eprosima::rpc::exception::BadParamException	This exception is thrown if the requested union member is not the current selection.

6.13.3.13 void FooDDS::Foo_FooProcedure_Result::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.14 FooDDS::Foo_FooProcedureCallbackHandler Class Reference

This abstract class defines the callbacks that eProsima RPC will call in an asynchronous call. These callback has to be implemented in a derived class.

`#include <FooDDSAsyncCallbackHandlers.h>`

Public Member Functions

- virtual void FooProcedure ()=0
- virtual void on_exception (const eprosima::rpc::exception::SystemException &ex)=0

  This function is called when an exception occurs. This exception can be launched in the server's side or in the client's side.

6.14.1 Detailed Description

This abstract class defines the callbacks that eProsima RPC will call in an asynchronous call. These callback has to be implemented in a derived class.

6.14.2 Member Function Documentation

6.14.2.1 virtual void FooDDS::Foo_FooProcedureCallbackHandler::FooProcedure ( ) [pure virtual]

This function is called when is received the reply from the server.

6.14.2.2 virtual void FooDDS::Foo_FooProcedureCallbackHandler::on_exception (const eprosima::rpc::exception::SystemException & *ex*) [pure virtual]

This function is called when an exception occurs. This exception can be launched in the server's side or in the client's side.

Parameters

ex	The exception that will be launched.

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

- utils/doxygen/examples/dds/FooDDSAsyncCallbackHandlers.h

6.15 FooDDS::Foo_FooProcedureTask Class Reference

This class represents a asynchronous task created to wait the reply of the procedure Foo::FooProcedure from the server in an asynchronous call.

`#include <FooDDSDDSAsyncSupport.h>`

Inheritance diagram for FooDDS::Foo_FooProcedureTask:

Collaboration diagram for FooDDS::Foo_FooProcedureTask:

Public Member Functions

- Foo_FooProcedureTask (Foo_FooProcedureCallbackHandler &obj)

  The default constructor.
- virtual ~Foo_FooProcedureTask ()

  Destructor.
- virtual void execute ()

  This funcion is called when the reply sample is received.
- virtual void on_exception (const eprosima::rpc::exception::SystemException &ex)

  This function is called when an exception occurs. This exception can be launched in the server's side or in the client's side.
- Foo_FooProcedureCallbackHandler & getObject ()

  This function returns the object used by the task.
- virtual void * getReplyInstance ()

  This function returns the allocated reply sample.

Additional Inherited Members

6.15.1 Detailed Description

This class represents a asynchronous task created to wait the reply of the procedure Foo::FooProcedure from the server in an asynchronous call.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 FooDDS::Foo_FooProcedureTask::Foo_FooProcedureTask (Foo_FooProcedureCallbackHandler & *obj*)

The default constructor.

Parameters

obj	Object that implements the callbacks that FastRPC will call when the reply will be received or and exception will be launched.
client	Pointer to the server's proxy. Cannot be NULL.

6.15.3 Member Function Documentation

6.15.3.1 Foo_FooProcedureCallbackHandler & FooDDS::Foo_FooProcedureTask::getObject ( )

This function returns the object used by the task.

Returns

The object that implements the callbacks.

6.15.3.2 void * FooDDS::Foo_FooProcedureTask::getReplyInstance ( ) [virtual]

This function returns the allocated reply sample.

Returns

Pointer to the allocated reply sample.

Implements eprosima::rpc::transport::dds::DDSAsyncTask.

6.15.3.3 void FooDDS::Foo_FooProcedureTask::on_exception (const eprosima::rpc::exception::SystemException & *ex*) [virtual]

This function is called when an exception occurs. This exception can be launched in the server's side or in the client's side.

Parameters

ex	The exception that will be launched.

Implements eprosima::rpc::transport::dds::DDSAsyncTask.

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

- utils/doxygen/examples/dds/FooDDSDDSAsyncSupport.h
- utils/doxygen/examples/dds/FooDDSDDSAsyncSupport.cxx

6.16 FooDDS::Foo_Reply Class Reference

This class represents the structure Foo_Reply that can be used to send/receive replies for the interface Foo.

`#include <FooDDSTopics.h>`

Public Member Functions

- user_cpp_DllExport Foo_Reply ()

  Default constructor.
- user_cpp_DllExport ~Foo_Reply ()

  Destructor.
- user_cpp_DllExport **Foo_Reply** (const Foo_Reply &x)
- user_cpp_DllExport Foo_Reply (Foo_Reply &&x)

  Move constructor.
- user_cpp_DllExport Foo_Reply & operator= (const Foo_Reply &x)

  Copy assignment.
- user_cpp_DllExport Foo_Reply & operator= (Foo_Reply &&x)

  Copy assignment.
- user_cpp_DllExport void header (const eprosima::rpc::protocol::dds::ReplyHeader &_header)

  This method sets the reply header information.
- user_cpp_DllExport void header (eprosima::rpc::protocol::dds::ReplyHeader &&_header)

  This method sets the reply header information.
- user_cpp_DllExport const eprosima::rpc::protocol::dds::ReplyHeader & header () const

  This method returns the reply header information.
- user_cpp_DllExport eprosima::rpc::protocol::dds::ReplyHeader & header ()

  This method returns the reply header information.
- user_cpp_DllExport void reply (const Foo_Return &_reply)

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport void reply (Foo_Return &&_reply)

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport const Foo_Return & reply () const

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport Foo_Return & reply ()

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const

  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)

  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.16.1 Detailed Description

This class represents the structure Foo_Reply that can be used to send/receive replies for the interface Foo.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 FooDDS::Foo_Reply::Foo_Reply (Foo_Reply && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_Reply that will be copied.

6.16.3 Member Function Documentation

6.16.3.1 void FooDDS::Foo_Reply::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.16.3.2 size_t FooDDS::Foo_Reply::getMaxCdrSerializedSize (size_t *current_alignment* = `0`) `[static]`

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.16.3.3 size_t FooDDS::Foo_Reply::getSerializedSize (size_t *current_alignment* = `0`) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.16.3.4 user_cpp_DllExport void FooDDS::Foo_Reply::header (const eprosima::rpc::protocol::dds::ReplyHeader & *_header*) `[inline]`

This method sets the reply header information.

Parameters

_header	Reply header.

6.16.3.5 user_cpp_DllExport void FooDDS::Foo_Reply::header (eprosima::rpc::protocol::dds::ReplyHeader && *_header*) [inline]

This method sets the reply header information.

Parameters

_header	Reply header.

6.16.3.6 user_cpp_DllExport const eprosima::rpc::protocol::dds::ReplyHeader& FooDDS::Foo_Reply::header ( ) const [inline]

This method returns the reply header information.

Returns

Reply header.

6.16.3.7 user_cpp_DllExport eprosima::rpc::protocol::dds::ReplyHeader& FooDDS::Foo_Reply::header ( ) [inline]

This method returns the reply header information.

Returns

Reply header.

6.16.3.8 FooDDS::Foo_Reply & FooDDS::Foo_Reply::operator= (const Foo_Reply & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo that will be copied.

6.16.3.9 FooDDS::Foo_Reply & FooDDS::Foo_Reply::operator= (Foo_Reply && *x*)

Copy assignment.

Parameters

x	Reference to the object Foo that will be copied.

6.16.3.10 user_cpp_DllExport void FooDDS::Foo_Reply::reply (const Foo_Return & *_reply*) [inline]

This method sets the union that encapsulates the interface operations.

Parameters

_reply	Union.

6.16.3.11 user_cpp_DllExport void FooDDS::Foo_Reply::reply (Foo_Return && *_reply*) [inline]

This method sets the union that encapsulates the interface operations.

Parameters

_reply	Union.

6.16.3.12 user_cpp_DllExport const Foo_Return& FooDDS::Foo_Reply::reply ( ) const [inline]

This method sets the union that encapsulates the interface operations.

Returns

Union.

6.16.3.13 user_cpp_DllExport Foo_Return& FooDDS::Foo_Reply::reply ( ) [inline]

This method sets the union that encapsulates the interface operations.

Returns

Union.

6.16.3.14 void FooDDS::Foo_Reply::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.17 FooDDS::Foo_ReplyDataReader Class Reference

Reply DataReader.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_ReplyDataReader:

Collaboration diagram for FooDDS::Foo_ReplyDataReader:

Public Member Functions

- **Foo_ReplyDataReader** (DDSDataReader ∗impl)

6.17.1 Detailed Description

Reply DataReader.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h

6.18 FooDDS::Foo_ReplyDataWriter Class Reference

Reply DataWriter.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_ReplyDataWriter:

Collaboration diagram for FooDDS::Foo_ReplyDataWriter:

Public Member Functions

- **Foo_ReplyDataWriter** (DDSDataWriter ∗impl)

6.18.1 Detailed Description

Reply DataWriter.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h

6.19 FooDDS::Foo_ReplyPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_ReplyPlugin:

Collaboration diagram for FooDDS::Foo_ReplyPlugin:

Public Member Functions

- DDSDataReader ∗ **create_datareaderI** (DDSDataReader ∗dataReader)
- DDS_ReturnCode_t **destroy_datareaderI** (DDSDataReader ∗dataReader)
- DDSDataWriter ∗ **create_datawriterI** (DDSDataWriter ∗dataWriter)
- DDS_ReturnCode_t **destroy_datawriterI** (DDSDataWriter ∗dataWriter)

Static Public Member Functions

- static const char ∗ **get_typename** ()
- static FooDDS::Foo_Reply ∗ **create_data** (void)
- static void **destroy_data** (FooDDS::Foo_Reply ∗sample)
- static void **copy_data** (FooDDS::Foo_Reply ∗dst, const FooDDS::Foo_Reply ∗src)
- static unsigned int **get_serialized_sample_max_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment)

- static unsigned int **get_serialized_sample_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment, const FooDDS::Foo_Reply *sample)
- static unsigned int **get_serialized_sample_min_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment)
- static PRESTypePluginParticipantData **on_participant_attached** (void *registration_data, const struct PRESTypePluginParticipantInfo *participant_info, RTIBool top_level_registration, void *container_plugin_context, RTICdrTypeCode *typeCode)
- static void **on_participant_detached** (PRESTypePluginParticipantData participant_data)
- static PRESTypePluginEndpointData **on_endpoint_attached** (PRESTypePluginParticipantData participant_data, const struct PRESTypePluginEndpointInfo *endpoint_info, RTIBool top_level_registration, void *container_plugin_context)
- static void **on_endpoint_detached** (PRESTypePluginEndpointData endpoint_data)
- static RTIBool **copy_sample** (PRESTypePluginEndpointData endpoint_data, FooDDS::Foo_Reply *dst, const FooDDS::Foo_Reply *src)
- static RTIBool **serialize** (PRESTypePluginEndpointData endpoint_data, const FooDDS::Foo_Reply *sample, struct RTICdrStream *stream, RTIBool serialize_encapsulation, RTIEncapsulationId encapsulation_id, RTIBool serialize_sample, void *endpoint_plugin_qos)
- static RTIBool **deserialize** (PRESTypePluginEndpointData endpoint_data, FooDDS::Foo_Reply **sample, RTIBool *drop_sample, struct RTICdrStream *stream, RTIBool deserialize_encapsulation, RTIBool deserialize_sample, void *endpoint_plugin_qos)
- static PRESTypePluginKeyKind **get_key_kind** (void)
- static DDS_TypeCode * **get_typecode** ()
- static struct PRESTypePlugin * **new_plugin** (void)
- static void **delete_plugin** (struct PRESTypePlugin *plugin)
- static bool register_type (DDSDomainParticipant *participant, const char *type_name)

6.19.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

6.19.2 Member Function Documentation

6.19.2.1 bool FooDDS::Foo_ReplyPlugin::register_type (DDSDomainParticipant * *participant*, const char * *type_name*) [static]

TODO Mover al transporte

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.20 FooDDS::Foo_Request Class Reference

This class represents the structure Foo_Request that can be used to send/receive requests for the interface Foo.

```
#include <FooDDSTopics.h>
```

Public Member Functions

- user_cpp_DllExport Foo_Request ()

  Default constructor.

- user_cpp_DllExport ~Foo_Request ()

  Destructor.
- user_cpp_DllExport Foo_Request (const Foo_Request &x)

  Copy constructor.
- user_cpp_DllExport Foo_Request (Foo_Request &&x)

  Move constructor.
- user_cpp_DllExport Foo_Request & operator= (const Foo_Request &x)

  Copy assignment.
- user_cpp_DllExport Foo_Request & operator= (Foo_Request &&x)

  Copy assignment.
- user_cpp_DllExport void header (const eprosima::rpc::protocol::dds::RequestHeader &_header)

  This method sets the request header information.
- user_cpp_DllExport void header (eprosima::rpc::protocol::dds::RequestHeader &&_header)

  This method sets the request header information.
- user_cpp_DllExport const eprosima::rpc::protocol::dds::RequestHeader & header () const

  This method returns the request header information.
- user_cpp_DllExport eprosima::rpc::protocol::dds::RequestHeader & header ()

  This method returns the request header information.
- user_cpp_DllExport void request (const Foo_Call &_request)

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport void request (Foo_Call &&_request)

  This method sets the union that encapsulates the interface operations.
- user_cpp_DllExport const Foo_Call & request () const

  This method returns the union that encapsulates the interface operations.
- user_cpp_DllExport Foo_Call & request ()

  This method returns the union that encapsulates the interface operations.
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const

  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)

  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.20.1 Detailed Description

This class represents the structure Foo_Request that can be used to send/receive requests for the interface Foo.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 FooDDS::Foo_Request::Foo_Request (const Foo_Request & *x*)

Copy constructor.

Parameters

x	Reference to the object Foo_Request that will be copied.

6.20.2.2 FooDDS::Foo_Request::Foo_Request (Foo_Request && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_Request that will be copied.

6.20.3 Member Function Documentation

6.20.3.1 void FooDDS::Foo_Request::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.20.3.2 size_t FooDDS::Foo_Request::getMaxCdrSerializedSize (size_t *current_alignment* = `0`) `[static]`

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.20.3.3 size_t FooDDS::Foo_Request::getSerializedSize (size_t *current_alignment* = `0`) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.20.3.4 user_cpp_DllExport void FooDDS::Foo_Request::header (const eprosima::rpc::protocol::dds::RequestHeader & *_header*) `[inline]`

This method sets the request header information.

Parameters

_header	Request header.

6.20.3.5 user_cpp_DllExport void FooDDS::Foo_Request::header (eprosima::rpc::protocol::dds::RequestHeader && *_header*) [inline]

This method sets the request header information.

Parameters

_header	Request header.

6.20.3.6 user_cpp_DllExport const eprosima::rpc::protocol::dds::RequestHeader& FooDDS::Foo_Request::header ( ) const [inline]

This method returns the request header information.

Returns

Request header.

6.20.3.7 user_cpp_DllExport eprosima::rpc::protocol::dds::RequestHeader& FooDDS::Foo_Request::header ( ) [inline]

This method returns the request header information.

Returns

Request header.

6.20.3.8 FooDDS::Foo_Request & FooDDS::Foo_Request::operator= (const Foo_Request & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo that will be copied.

6.20.3.9 FooDDS::Foo_Request & FooDDS::Foo_Request::operator= (Foo_Request && *x*)

Copy assignment.

Parameters

x	Reference to the object Foo that will be copied.

6.20.3.10 user_cpp_DllExport void FooDDS::Foo_Request::request (const Foo_Call & *_request*) [inline]

This method sets the union that encapsulates the interface operations.

Parameters

_request	Union.

6.20.3.11 user_cpp_DllExport void FooDDS::Foo_Request::request (Foo_Call && *_request*) [inline]

This method sets the union that encapsulates the interface operations.

Parameters

_request	Union.

6.20.3.12 user_cpp_DllExport const Foo_Call& FooDDS::Foo_Request::request ( ) const [inline]

This method returns the union that encapsulates the interface operations.

Returns

Union.

6.20.3.13 user_cpp_DllExport Foo_Call& FooDDS::Foo_Request::request ( ) [inline]

This method returns the union that encapsulates the interface operations.

Returns

Union.

6.20.3.14 void FooDDS::Foo_Request::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.21 FooDDS::Foo_RequestDataReader Class Reference

Request DataReader.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_RequestDataReader:

Collaboration diagram for FooDDS::Foo_RequestDataReader:

Public Member Functions

- **Foo_RequestDataReader** (DDSDataReader ∗impl)

6.21.1 Detailed Description

Request DataReader.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h

6.22 FooDDS::Foo_RequestDataWriter Class Reference

Request DataWriter.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_RequestDataWriter:

Collaboration diagram for FooDDS::Foo_RequestDataWriter:

Public Member Functions

- **Foo_RequestDataWriter** (DDSDataWriter ∗impl)

6.22.1 Detailed Description

Request DataWriter.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h

6.23 FooDDS::Foo_RequestPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::Foo_RequestPlugin:

Collaboration diagram for FooDDS::Foo_RequestPlugin:

Public Member Functions

- DDSDataReader ∗ **create_datareaderI** (DDSDataReader ∗dataReader)
- DDS_ReturnCode_t **destroy_datareaderI** (DDSDataReader ∗dataReader)
- DDSDataWriter ∗ **create_datawriterI** (DDSDataWriter ∗dataWriter)
- DDS_ReturnCode_t **destroy_datawriterI** (DDSDataWriter ∗dataWriter)

Static Public Member Functions

- static const char ∗ **get_typename** ()
- static FooDDS::Foo_Request ∗ **create_data** (void)
- static void **destroy_data** (FooDDS::Foo_Request ∗sample)
- static void **copy_data** (FooDDS::Foo_Request ∗dst, const FooDDS::Foo_Request ∗src)
- static unsigned int **get_serialized_sample_max_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment)

- static unsigned int **get_serialized_sample_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment, const FooDDS::Foo_Request *sample)
- static unsigned int **get_serialized_sample_min_size** (PRESTypePluginEndpointData endpoint_data, RTIBool include_encapsulation, RTIEncapsulationId encapsulation_id, unsigned int current_alignment)
- static PRESTypePluginParticipantData **on_participant_attached** (void *registration_data, const struct PRESTypePluginParticipantInfo *participant_info, RTIBool top_level_registration, void *container_plugin_context, RTICdrTypeCode *typeCode)
- static void **on_participant_detached** (PRESTypePluginParticipantData participant_data)
- static PRESTypePluginEndpointData **on_endpoint_attached** (PRESTypePluginParticipantData participant_data, const struct PRESTypePluginEndpointInfo *endpoint_info, RTIBool top_level_registration, void *container_plugin_context)
- static void **on_endpoint_detached** (PRESTypePluginEndpointData endpoint_data)
- static RTIBool **copy_sample** (PRESTypePluginEndpointData endpoint_data, FooDDS::Foo_Request *dst, const FooDDS::Foo_Request *src)
- static RTIBool **serialize** (PRESTypePluginEndpointData endpoint_data, const FooDDS::Foo_Request *sample, struct RTICdrStream *stream, RTIBool serialize_encapsulation, RTIEncapsulationId encapsulation_id, RTIBool serialize_sample, void *endpoint_plugin_qos)
- static RTIBool **deserialize** (PRESTypePluginEndpointData endpoint_data, FooDDS::Foo_Request **sample, RTIBool *drop_sample, struct RTICdrStream *stream, RTIBool deserialize_encapsulation, RTIBool deserialize_sample, void *endpoint_plugin_qos)
- static PRESTypePluginKeyKind **get_key_kind** (void)
- static DDS_TypeCode * **get_typecode** ()
- static struct PRESTypePlugin * **new_plugin** (void)
- static void **delete_plugin** (struct PRESTypePlugin *plugin)
- static bool register_type (DDSDomainParticipant *participant, const char *type_name)

6.23.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

6.23.2 Member Function Documentation

6.23.2.1 bool FooDDS::Foo_RequestPlugin::register_type (DDSDomainParticipant * *participant,* const char * *type_name*) [static]

TODO Mover al transporte

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.24 FooDDS::Foo_Return Class Reference

This class represents the union used in the DDS topic to encapsulate the operations in reply samples.

```
#include <FooDDSTopics.h>
```

Public Member Functions

- user_cpp_DllExport Foo_Return ()

  Default constructor.

- user_cpp_DllExport ~Foo_Return ()
  Destructor.
- user_cpp_DllExport Foo_Return (const Foo_Return &x)
  Copy constructor.
- user_cpp_DllExport Foo_Return (Foo_Return &&x)
  Move constructor.
- user_cpp_DllExport Foo_Return & operator= (const Foo_Return &x)
  Copy assignment.
- user_cpp_DllExport Foo_Return & operator= (Foo_Return &&x)
  Move assignment.
- user_cpp_DllExport void _d (int32_t __d)
  This function sets the discriminator value.
- user_cpp_DllExport int32_t _d () const
  This function returns the value of the discriminator.
- user_cpp_DllExport int32_t & _d ()
  This function returns a reference to the discriminator.
- user_cpp_DllExport void **unknown_operation** (eprosima::rpc::protocol::dds::UnknownOperation _unknown_operation)
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownOperation **unknown_operation** () const
- user_cpp_DllExport eprosima::rpc::protocol::dds::UnknownOperation & **unknown_operation** ()
- user_cpp_DllExport void FooProcedure (const Foo_FooProcedure_Result &_FooProcedure)
  This function copies the value in member FooProcedure.
- user_cpp_DllExport void FooProcedure (Foo_FooProcedure_Result &&_FooProcedure)
  This function moves the value in member FooProcedure.
- user_cpp_DllExport const Foo_FooProcedure_Result & FooProcedure () const
  This function returns a constant reference to member FooProcedure.
- user_cpp_DllExport Foo_FooProcedure_Result & FooProcedure ()
  This function returns a reference to member FooProcedure.
- user_cpp_DllExport size_t getSerializedSize (size_t current_alignment=0) const
  This function returns the serialized size of an object depending on the buffer alignment.
- user_cpp_DllExport void serialize (eprosima::fastcdr::Cdr &cdr) const
  This function serializes an object using CDR serialization.
- user_cpp_DllExport void deserialize (eprosima::fastcdr::Cdr &cdr)
  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static user_cpp_DllExport size_t getMaxCdrSerializedSize (size_t current_alignment=0)
  This function returns the maximum serialized size of an object depending on the buffer alignment.

6.24.1 Detailed Description

This class represents the union used in the DDS topic to encapsulate the operations in reply samples.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 FooDDS::Foo_Return::Foo_Return (const Foo_Return & *x*)

Copy constructor.

Parameters

x	Reference to the object Foo_Return that will be copied.

6.24.2.2 FooDDS::Foo_Return::Foo_Return (Foo_Return && *x*)

Move constructor.

Parameters

x	Reference to the object Foo_Return that will be copied.

6.24.3 Member Function Documentation

6.24.3.1 void FooDDS::Foo_Return::_d (int32_t *__d*)

This function sets the discriminator value.

Parameters

__d	New value for the discriminator.

Exceptions

eprosima::rpc::exception::BadParamException	This exception is thrown if the new value doesn't correspond to the selected union member.

6.24.3.2 int32_t FooDDS::Foo_Return::_d ( ) const

This function returns the value of the discriminator.

Returns

Value of the discriminator

6.24.3.3 int32_t & FooDDS::Foo_Return::_d ( )

This function returns a reference to the discriminator.

Returns

Reference to the discriminator.

6.24.3.4 void FooDDS::Foo_Return::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.24.3.5 void FooDDS::Foo_Return::FooProcedure (const Foo_FooProcedure_Result & *_FooProcedure*)

This function copies the value in member FooProcedure.

Parameters

_FooProcedure	New value to be copied in member FooProcedure

6.24.3.6 void FooDDS::Foo_Return::FooProcedure (FooDDS::Foo_FooProcedure_Result && *_FooProcedure*)

This function moves the value in member FooProcedure.

Parameters

_FooProcedure	New value to be moved in member FooProcedure

6.24.3.7 const FooDDS::Foo_FooProcedure_Result & FooDDS::Foo_Return::FooProcedure ( ) const

This function returns a constant reference to member FooProcedure.

Returns

Constant reference to member FooProcedure

6.24.3.8 FooDDS::Foo_FooProcedure_Result & FooDDS::Foo_Return::FooProcedure ( )

This function returns a reference to member FooProcedure.

Returns

Reference to member FooProcedure

6.24.3.9 size_t FooDDS::Foo_Return::getMaxCdrSerializedSize (size_t *current_alignment* = `0`) `[static]`

This function returns the maximum serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.24.3.10 size_t FooDDS::Foo_Return::getSerializedSize (size_t *current_alignment* = `0`) const

This function returns the serialized size of an object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Serialized size.

6.24.3.11 FooDDS::Foo_Return & FooDDS::Foo_Return::operator= (const Foo_Return & *x*)

Copy assignment.

Parameters

x	Reference to the object Foo_Return that will be copied.

6.24.3.12 FooDDS::Foo_Return & FooDDS::Foo_Return::operator= (Foo_Return && *x*)

Move assignment.

Parameters

x	Reference to the object Foo_Return that will be copied.

6.24.3.13 void FooDDS::Foo_Return::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

6.25 FooDDS::Foo_ReturnPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * **get_typecode** ()

6.25.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.26 eprosima::rpc::protocol::dds::FooDDSProtocol Class Reference

This class is responsible for serializing and deserializing the requests and responses of this application. It uses DDS.

`#include <FooDDSDDSProtocol.h>`

Inheritance diagram for eprosima::rpc::protocol::dds::FooDDSProtocol:

Collaboration diagram for eprosima::rpc::protocol::dds::FooDDSProtocol:

Public Member Functions

- FooDDSProtocol ()

  Default constructor.
- virtual ~FooDDSProtocol ()

  Destructor.
- virtual bool setTransport (eprosima::rpc::transport::Transport &transport)

  This method sets the transport for the communications.
- bool activateInterface (const char *interfaceName)

  This function activates needed DDS entities to use an interface.
- void FooDDS_Foo_FooProcedure ()

  This method implements the proxy part of the protocol for the operation FooProcedure. It is called from the Proxy interface.
- void FooDDS_Foo_FooProcedure_async (FooDDS::Foo_FooProcedureCallbackHandler &obj)

  This asynchronous method implements the proxy part of the protocol for the operation FooProcedure. It is called from the Proxy interface.

Static Public Member Functions

- static void FooDDS_Foo_serve (eprosima::rpc::protocol::Protocol &protocol, void *data, eprosima::rpc::transport::Endpoint *endpoint)

  This method implements the server part of the protocol for the interface Foo. It is called when a request sample is received.

Additional Inherited Members

6.26.1 Detailed Description

This class is responsible for serializing and deserializing the requests and responses of this application. It uses DDS.

6.26.2 Member Function Documentation

6.26.2.1 bool FooDDSProtocol::activateInterface (const char * *interfaceName*) [virtual]

This function activates needed DDS entities to use an interface.

Parameters

interfaceName	Interface name.

Returns

Whether the activation works successfully.

Implements eprosima::rpc::protocol::FooDDSProtocol.

6.26.2.2 void FooDDSProtocol::FooDDS_Foo_serve (eprosima::rpc::protocol::Protocol & *protocol*, void * *data*, eprosima::rpc::transport::Endpoint * *endpoint*) [static]

This method implements the server part of the protocol for the interface Foo. It is called when a request sample is received.

Parameters

protocol	DDS protocol object that is in used.
data	Pointer to the received request sample. Cannot be NULL.
endpoint	Pointer to the endpoint that sent the request reply. Cannot be NULL.

6.26.2.3 virtual bool eprosima::rpc::protocol::dds::FooDDSProtocol::setTransport (eprosima::rpc::transport::Transport & *transport*) [virtual]

This method sets the transport for the communications.

Parameters

transport	Transport to use

Returns

True if the assignment is successful, false otherwise

Implements eprosima::rpc::protocol::FooDDSProtocol.

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

- utils/doxygen/examples/dds/FooDDSDDSProtocol.h
- utils/doxygen/examples/dds/FooDDSDDSProtocol.cxx

6.27 eprosima::rpc::protocol::FooDDSProtocol Class Reference

Protocol base class for the specific application.

`#include <FooDDSProtocol.h>`

Inheritance diagram for eprosima::rpc::protocol::FooDDSProtocol:

Collaboration diagram for eprosima::rpc::protocol::FooDDSProtocol:

Public Member Functions

- virtual bool setTransport (eprosima::rpc::transport::Transport &transport)=0

  This method sets the transport for the communications. It has to be implemented by the children classes.
- virtual bool activateInterface (const char *interfaceName)=0

  In some protocols this function activates needed entities to use an interface.
- void linkFooDDS_FooImpl (FooDDS::FooServerImpl &impl)

  This method links a specific servant with the protocol.
- virtual void FooDDS_Foo_FooProcedure ()=0

  This method implements the proxy part of the protocol for the operation FooProcedure. It has to be implemented by the child classes.
- virtual void FooDDS_Foo_FooProcedure_async (FooDDS::Foo_FooProcedureCallbackHandler &obj)=0

  This asynchronous method implements the proxy part of the protocol for the operation FooProcedure. It has to be implemented by the child classes.

Protected Attributes

- FooDDS::FooServerImpl * **_FooDDS_Foo_impl**

Additional Inherited Members

6.27.1 Detailed Description

Protocol base class for the specific application.

6.27.2 Member Function Documentation

6.27.2.1 virtual bool eprosima::rpc::protocol::FooDDSProtocol::activateInterface (const char * *interfaceName*) [pure virtual]

In some protocols this function activates needed entities to use an interface.

Parameters

interfaceName	Interface name.

Returns

Whether the activation works successfully.

Implemented in eprosima::rpc::protocol::dds::FooDDSProtocol.

6.27.2.2 void eprosima::rpc::protocol::FooDDSProtocol::linkFooDDS_FooImpl (FooDDS::FooServerImpl & *impl*) [inline]

This method links a specific servant with the protocol.

Parameters

impl	Servant implementation.

6.27.2.3 virtual bool eprosima::rpc::protocol::FooDDSProtocol::setTransport (eprosima::rpc::transport::Transport & *transport*) [pure virtual]

This method sets the transport for the communications. It has to be implemented by the children classes.

Parameters

transport	Transport to use.

Returns

True if the assignment is successful, false otherwise

Implements eprosima::rpc::protocol::Protocol.

Implemented in eprosima::rpc::protocol::dds::FooDDSProtocol.

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

- utils/doxygen/examples/dds/FooDDSProtocol.h
- utils/doxygen/examples/dds/FooDDSDDSProtocol.cxx

6.28 FooDDS::FooPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Classes

- class FooProcedure_InPlugin

  This class encapsulates the methods used on DDS topics by DDS middleware.
- class FooProcedure_OutPlugin

  This class encapsulates the methods used on DDS topics by DDS middleware.
- class FooProcedure_ResultPlugin

  This class encapsulates the methods used on DDS topics by DDS middleware.

6.28.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h

6.29 FooDDS::FooPlugin::FooProcedure_InPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * **get_typecode** ()

6.29.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.30 FooDDS::FooPlugin::FooProcedure_OutPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * **get_typecode** ()

6.30.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.31 FooDDS::FooPlugin::FooProcedure_ResultPlugin Class Reference

This class encapsulates the methods used on DDS topics by DDS middleware.

`#include <FooDDSTopicsPlugin.h>`

Inheritance diagram for FooDDS::FooPlugin::FooProcedure_ResultPlugin:

Collaboration diagram for FooDDS::FooPlugin::FooProcedure_ResultPlugin:

Static Public Member Functions

- static DDS_TypeCode * **get_typecode** ()

6.31.1 Detailed Description

This class encapsulates the methods used on DDS topics by DDS middleware.

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

- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

6.32 FooDDS::FooProxy Class Reference

This class implements a specific server's proxy for the defined interface Foo.

`#include <FooDDSProxy.h>`

Inheritance diagram for FooDDS::FooProxy:

Collaboration diagram for FooDDS::FooProxy:

Public Member Functions

- FooProxy (eprosima::rpc::transport::ProxyTransport &transport, eprosima::rpc::protocol::FooDDSProtocol &protocol)

  This constructor sets the transport that will be used by the server's proxy.
- virtual ~FooProxy ()

  Destructor.
- void FooProcedure ()

  Proxy method for the operation FooProcedure.
- void FooProcedure_async (FooDDS::Foo_FooProcedureCallbackHandler &obj)

  Proxy asynchronous method for the operation FooProcedure.

Additional Inherited Members

6.32.1 Detailed Description

This class implements a specific server's proxy for the defined interface Foo.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 FooDDS::FooProxy::FooProxy (eprosima::rpc::transport::ProxyTransport & *transport,* eprosima::rpc::protocol::FooDDSProtocol & *protocol*)

This constructor sets the transport that will be used by the server's proxy.

Parameters

transport	The network transport that server's proxy has to use. This transport's object is not deleted by this class in its destructor. Cannot be NULL.
protocol	The protocol used to send the information over the transport. This protocol's object is not deleted by this class in its destructor. Cannot be NULL.

Exceptions

eprosima::rpc::exception::InitializeException	This exception is thrown when the initialization was wrong.

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

- utils/doxygen/examples/dds/FooDDSProxy.h
- utils/doxygen/examples/dds/FooDDSProxy.cxx

6.33 FooDDS::FooServer Class Reference

This class implements a specific server for the defined interface Foo by user.

`#include <FooDDSServer.h>`

Inheritance diagram for FooDDS::FooServer:

Collaboration diagram for FooDDS::FooServer:

Public Member Functions

- FooServer (eprosima::rpc::strategy::ServerStrategy &strategy, eprosima::rpc::transport::ServerTransport &transport, eprosima::rpc::protocol::FooDDSProtocol &protocol, FooServerImpl &servant)

  This constructor sets the transport that will be used by the server.
- virtual ~FooServer ()

  Destructor.

Additional Inherited Members

6.33.1 Detailed Description

This class implements a specific server for the defined interface Foo by user.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 FooDDS::FooServer::FooServer (eprosima::rpc::strategy::ServerStrategy & *strategy,* eprosima::rpc::transport::ServerTransport & *transport,* eprosima::rpc::protocol::FooDDSProtocol & *protocol,* FooServerImpl & *servant*)

This constructor sets the transport that will be used by the server.

Parameters

strategy	Strategy used by server to work with new requests. This class doesn't delete this object in its destructor. Cannot be NULL.

transport	The network transport that the server has to use. This transport's object is not deleted by this class in its destructor. Cannot be NULL.
protocol	Generated protocol that the server has to use. This class has the information to process requests and build responses for this application environment.
servant	Servant that the server will use to invoke user's functions.

Exceptions

eProsima::RPCDDS::InitializeException	This exception is thrown when the initialization was wrong.

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

- utils/doxygen/examples/dds/FooDDSServer.h
- utils/doxygen/examples/dds/FooDDSServer.cxx

6.34 FooDDS::FooServerImpl Class Reference

This class is the skeleton of the servant and its remote procedures has to be implemented.

`#include <FooDDSServerImpl.h>`

Inheritance diagram for FooDDS::FooServerImpl:

Collaboration diagram for FooDDS::FooServerImpl:

Public Member Functions

- FooServerImpl ()

  The default constructor.
- virtual ~FooServerImpl ()

  Destructor.

6.34.1 Detailed Description

This class is the skeleton of the servant and its remote procedures has to be implemented.

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

- utils/doxygen/examples/dds/FooDDSServerImpl.h

6.35 eprosima::rpc::protocol::dds::GUID_t Class Reference

Public Member Functions

- **GUID_t** (const GUID_t &guid)
- GUID_t & **operator=** (const GUID_t &guid)
- uint8_t * **value** ()
- void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes the GUID_t object using CDR serialization.
- void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes the GUID_t object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment)

  This function returns the maximum serialized size of a GUID_t object depending on the buffer alignment.

6.35.1 Member Function Documentation

6.35.1.1 void eprosima::rpc::protocol::dds::GUID_t::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes the GUID_t object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.35.1.2 static size_t eprosima::rpc::protocol::dds::GUID_t::getMaxCdrSerializedSize (size_t *current_alignment*) [static]

This function returns the maximum serialized size of a GUID_t object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.35.1.3 void eprosima::rpc::protocol::dds::GUID_t::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes the GUID_t object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- includetmp/rpcdds/protocols/dds/MessageHeader.h

6.36 eprosima::rpc::protocol::dds::GUID_tPlugin Class Reference

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()

  This function returns the TypeCode.

6.36.1 Member Function Documentation

6.36.1.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::GUID_tPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.37 eprosima::rpc::exception::IncompatibleException Class Reference

This class is thrown as an exception when a selected protocol and transport are incompatible.

`#include <IncompatibleException.h>`

Inheritance diagram for eprosima::rpc::exception::IncompatibleException:

Collaboration diagram for eprosima::rpc::exception::IncompatibleException:

Public Member Functions

- FASTRPC_DllAPI IncompatibleException (const std::string &message)
  Default constructor.
- FASTRPC_DllAPI IncompatibleException (const IncompatibleException &ex)
  Default copy constructor.
- FASTRPC_DllAPI IncompatibleException (IncompatibleException &&ex)
  Default move constructor.
- FASTRPC_DllAPI IncompatibleException & operator= (const IncompatibleException &ex)
  Assigment operation.
- FASTRPC_DllAPI IncompatibleException & operator= (IncompatibleException &&ex)
  Assigment operation.
- virtual FASTRPC_DllAPI ~IncompatibleException () throw ()
  Default constructor.
- virtual FASTRPC_DllAPI void raise () const
  This function throws the object as an exception.

Additional Inherited Members

6.37.1 Detailed Description

This class is thrown as an exception when a selected protocol and transport are incompatible.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 FASTRPC_DllAPI eprosima::rpc::exception::IncompatibleException::IncompatibleException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.37.2.2 FASTRPC_DllAPI eprosima::rpc::exception::IncompatibleException::IncompatibleException (const IncompatibleException & *ex*)

Default copy constructor.

Parameters

ex	IncompatibleException that will be copied.

6.37.2.3 FASTRPC_DllAPI eprosima::rpc::exception::IncompatibleException::IncompatibleException (IncompatibleException && *ex*)

Default move constructor.

Parameters

ex	IncompatibleException that will be moved.

6.37.3 Member Function Documentation

6.37.3.1 FASTRPC_DllAPI IncompatibleException& eprosima::rpc::exception::IncompatibleException::operator= (const IncompatibleException & *ex*)

Assigment operation.

Parameters

ex	IncompatibleException that will be copied.

6.37.3.2 FASTRPC_DllAPI IncompatibleException& eprosima::rpc::exception::IncompatibleException::operator= (IncompatibleException && *ex*)

Assigment operation.

Parameters

ex	IncompatibleException that will be moved.

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

- includetmp/rpcdds/exceptions/IncompatibleException.h

6.38 eprosima::rpc::exception::InitializeException Class Reference

This class is thrown as an exception when there is an error initializating an object.

`#include <InitializeException.h>`

Inheritance diagram for eprosima::rpc::exception::InitializeException:

Collaboration diagram for eprosima::rpc::exception::InitializeException:

Public Member Functions

- FASTRPC_DllAPI InitializeException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI InitializeException (const InitializeException &ex)

  Default copy constructor.
- FASTRPC_DllAPI InitializeException (InitializeException &&ex)

  Default move constructor.
- FASTRPC_DllAPI InitializeException & operator= (const InitializeException &ex)

  Assigment operation.
- FASTRPC_DllAPI InitializeException & operator= (InitializeException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~InitializeException () throw ()

  Default constructor.
- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.38.1 Detailed Description

This class is thrown as an exception when there is an error initializating an object.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 FASTRPC_DllAPI eprosima::rpc::exception::InitializeException::InitializeException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.38.2.2 FASTRPC_DllAPI eprosima::rpc::exception::InitializeException::InitializeException (const InitializeException & *ex*)

Default copy constructor.

Parameters

ex	InitializeException that will be copied.

6.38.2.3 FASTRPC_DllAPI eprosima::rpc::exception::InitializeException::InitializeException (InitializeException && *ex*)

Default move constructor.

Parameters

ex	InitializeException that will be moved.

6.38.3 Member Function Documentation

6.38.3.1 FASTRPC_DllAPI InitializeException& eprosima::rpc::exception::InitializeException::operator= (const InitializeException & *ex*)

Assigment operation.

Parameters

ex	InitializeException that will be copied.

6.38.3.2 FASTRPC_DllAPI InitializeException& eprosima::rpc::exception::InitializeException::operator= (InitializeException && *ex*)

Assigment operation.

Parameters

ex	InitializeException that will be moved.

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

- includetmp/rpcdds/exceptions/InitializeException.h

6.39 eprosima::rpc::protocol::Protocol Class Reference

This abstract class represents the protocol used by the RPCs. It serializes and deserializes the information and uses a eprosima::rpc::transport::Transport to send it and receive it.

`#include <Protocol.h>`

Inheritance diagram for eprosima::rpc::protocol::Protocol:

Public Member Functions

- virtual bool setTransport (eprosima::rpc::transport::Transport &transport)=0

  This method sets a eprosima::rpc::transport::Transport object, used for the communications.

Protected Member Functions

- Protocol ()

  Default constructor.
- virtual ~Protocol ()

  Default destructor.
- eprosima::rpc::transport::Transport & getTransport () const

  This method returns the eprosima::rpc::transport::Transport object, used for the communications.
- void _setTransport (eprosima::rpc::transport::Transport &transport)

  This method sets a eprosima::rpc::transport::Transport object, used for the communications.

6.39.1 Detailed Description

This abstract class represents the protocol used by the RPCs. It serializes and deserializes the information and uses a eprosima::rpc::transport::Transport to send it and receive it.

6.39.2 Member Function Documentation

6.39.2.1 void eprosima::rpc::protocol::Protocol::_setTransport (eprosima::rpc::transport::Transport & *transport*) [inline], [protected]

This method sets a eprosima::rpc::transport::Transport object, used for the communications.

Parameters

transport	eprosima::rpc::transport::Transport to use for the communication.

6.39.2.2 eprosima::rpc::transport::Transport& eprosima::rpc::protocol::Protocol::getTransport ( ) const [inline], [protected]

This method returns the eprosima::rpc::transport::Transport object, used for the communications.

Returns

eprosima::rpc::transport::Transport used for the communications.

6.39.2.3 virtual bool eprosima::rpc::protocol::Protocol::setTransport (eprosima::rpc::transport::Transport & *transport*) [pure virtual]

This method sets a eprosima::rpc::transport::Transport object, used for the communications.

Parameters

transport	eprosima::rpc::transport::Transport to use for the communications.

Implemented in eprosima::rpc::protocol::dds::FooDDSProtocol, and eprosima::rpc::protocol::FooDDSProtocol.

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

- includetmp/rpcdds/protocols/Protocol.h

6.40 eprosima::rpc::proxy::Proxy Class Reference

This class implements the common functionalities that all server's proxies have.

`#include <Proxy.h>`

Inheritance diagram for eprosima::rpc::proxy::Proxy:

Protected Member Functions

- Proxy (eprosima::rpc::transport::ProxyTransport &transport, eprosima::rpc::protocol::Protocol &protocol)
  Proxy constructor.
- virtual ~Proxy ()
  The default destructor.
- eprosima::rpc::protocol::Protocol & getProtocol () const

Method to obtain the protocol.

- eprosima::rpc::transport::ProxyTransport & getTransport () const

  Method to get the transport.

6.40.1 Detailed Description

This class implements the common functionalities that all server's proxies have.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 eprosima::rpc::proxy::Proxy::Proxy (eprosima::rpc::transport::ProxyTransport & *transport*, eprosima::rpc::protocol::Protocol & *protocol*) [protected]

Proxy constructor.

Parameters

transport	The transport that will be used by the server's proxy. This class doesn't delete this object in its destructor.
protocol	The protocol used to send information over the transport. This class doesn't delete this object in its destructor.

Exceptions

InitializeException	This exception is thrown when the initialization went wrong.

6.40.3 Member Function Documentation

6.40.3.1 eprosima::rpc::protocol::Protocol& eprosima::rpc::proxy::Proxy::getProtocol ( ) const [inline], [protected]

Method to obtain the protocol.

Returns

The protocol used to send information over the transport

6.40.3.2 eprosima::rpc::transport::ProxyTransport& eprosima::rpc::proxy::Proxy::getTransport ( ) const [inline], [protected]

Method to get the transport.

Returns

The transport used used by the proxy

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

- includetmp/rpcdds/client/Proxy.h

6.41 eprosima::rpc::transport::dds::ProxyProcedureEndpoint Class Reference

This class represents a remote endpoint used by a proxy. It also encapsulates the DDS datawriter and the DDS datareader.

`#include <ProxyProcedureEndpoint.h>`

Inheritance diagram for eprosima::rpc::transport::dds::ProxyProcedureEndpoint:

Collaboration diagram for eprosima::rpc::transport::dds::ProxyProcedureEndpoint:

Public Member Functions

- ProxyProcedureEndpoint (ProxyTransport &transport)

  Default constructor.
- virtual ~ProxyProcedureEndpoint ()

  Default destructor.
- int initialize (const char *name, const char *writertypename, const char *readertypename, bool eprosima_types, Transport::Copy_data copy_data, int dataSize)

  This function initializes the proxy procedure endpoint.
- void finalize ()

  This function finalizes the proxy procedure endpoint. All entities and objects created by this procedure endpoint are deleted.
- eprosima::rpc::ReturnMessage send (void *request, void *reply)

  This function sends a synchronous RPC call. It sends the request to the server and waits for the reply. The wait mechanism is implemented with a DDS WaitSet.
- eprosima::rpc::ReturnMessage send_async (void *request, DDSAsyncTask *task)

  This function sends an asynchronous RPC call. It sends the request to the server and does not wait for the reply. Instead, the corresponding callback inside the DDSAsyncTask object will be invoked when the response arrives.
- void freeQuery (DDSQueryCondition *query)

  Frees a DDS query condition.
- eprosima::rpc::ReturnMessage takeReply (void *reply, DDSQueryCondition *query)

  This function takes a sample from the datareader.

Additional Inherited Members

6.41.1 Detailed Description

This class represents a remote endpoint used by a proxy. It also encapsulates the DDS datawriter and the DDS datareader.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 eprosima::rpc::transport::dds::ProxyProcedureEndpoint::ProxyProcedureEndpoint (ProxyTransport & *transport*)

Default constructor.

Parameters

Transport	that is creating the proxy procedure endpoint. It cannot be NULL.

6.41.3 Member Function Documentation

6.41.3.1 void eprosima::rpc::transport::dds::ProxyProcedureEndpoint::freeQuery (DDSQueryCondition * *query*)

Frees a DDS query condition.

Parameters

query	Query condition to free.

6.41.3.2 int eprosima::rpc::transport::dds::ProxyProcedureEndpoint::initialize (const char * *name,* const char * *writertypename,* const char * *readertypename,* bool *eprosima_types,* Transport::Copy_data *copy_data,* int *dataSize*)

This function initializes the proxy procedure endpoint.

Parameters

name	The name associated with this proxy procedure endpoint. It cannot be NULL.
writertypename	The type name of the topic that the proxy procedure endpoint uses in the datawriter. It cannot be NULL.
readertypename	The type name of the topic that the proxy procedure endpoint uses in the datareader. It cannot be NULL.
copy_data	Pointer to the function used to copy the data when it is received.

Returns

0 if the initialization works. -1 in other case. TODO

6.41.3.3 eprosima::rpc::ReturnMessage eprosima::rpc::transport::dds::ProxyProcedureEndpoint::send (void * *request,* void * *reply*)

This function sends a synchronous RPC call. It sends the request to the server and waits for the reply. The wait mechanism is implemented with a DDS WaitSet.

Parameters

request	Pointer to the allocated request. It cannot be NULL.
reply	Pointer to the allocated reply. This memory will be filled with the incoming data. The pointer can be NULL and this means that the RPC call is oneway.

Returns

Operation status

Exceptions

eprosima::rpc::exception::-ServerTimeoutException.	

6.41.3.4 eprosima::rpc::ReturnMessage eprosima::rpc::transport::dds::ProxyProcedureEndpoint::send_async (void * *request,* DDSAsyncTask * *task*)

This function sends an asynchronous RPC call. It sends the request to the server and does not wait for the reply. Instead, the corresponding callback inside the DDSAsyncTask object will be invoked when the response arrives.

Parameters

request	Pointer to the allocated request. It cannot be NULL.

task	Object containing information of the asynchronous task.

Returns

Operation status. It can be CLIENT_INTERNAL_ERROR or NO_SERVER

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

- includetmp/rpcdds/transports/dds/components/ProxyProcedureEndpoint.h

6.42 eprosima::rpc::transport::dds::ProxyTransport Class Reference

This class is the base of all proxies that implement a transport using DDS.

`#include <ProxyTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::ProxyTransport:

Collaboration diagram for eprosima::rpc::transport::dds::ProxyTransport:

Public Member Functions

- virtual FASTRPC_DllAPI ~ProxyTransport ()

  Default destructor.
- virtual FASTRPC_DllAPI const char * getType () const

  This abstract function returns the type of the transport. This function has to be implemented by the child classes.
- FASTRPC_DllAPI const char * getRemoteServiceName () const

  This function returns the DDS service name.
- FASTRPC_DllAPI const char * **getInstanceName** () const
- FASTRPC_DllAPI long getTimeout ()

  This function gets the timeout value.
- FASTRPC_DllAPI eprosima::rpc::transport::Endpoint * createProcedureEndpoint (const char *name, const char *writertypename, const char *readertypename, bool eprosima_types, Transport::Create_data create_data, Transport::Copy_data copy_data, Transport::Destroy_data destroy_data, Transport::ProcessFunc processFunc, int dataSize)

  This function creates a new proxy procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.
- FASTRPC_DllAPI int addAsyncTask (DDSQueryCondition *query, DDSAsyncTask *task, long timeout)

  This function adds a asynchronous task to the asynchronous thread.
- FASTRPC_DllAPI void deleteAssociatedAsyncTasks (ProxyProcedureEndpoint *pe)

  This function deletes all the asynchronous tasks associated with the ProxyProcedureEndpoint endpoint.

Protected Member Functions

- virtual FASTRPC_DllAPI int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant *participant)=0

  This abstract function sets the QoS to use a specific transport.
- FASTRPC_DllAPI ProxyTransport (const char *const &remoteServiceName, const char *const &instanceName, int domainId=0, long milliseconds=10000L)

  Default constructor.

Additional Inherited Members

6.42.1 Detailed Description

This class is the base of all proxies that implement a transport using DDS.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::ProxyTransport::ProxyTransport (const char *const & *remoteServiceName,* const char *const & *instanceName,* int *domainId* = 0, long *milliseconds* = 10000L) [protected]

Default constructor.

Parameters

domainId	Optional parameter that specifies the domain identifier will be used in DDS.

6.42.3 Member Function Documentation

6.42.3.1 FASTRPC_DllAPI int eprosima::rpc::transport::dds::ProxyTransport::addAsyncTask (DDSQueryCondition * *query,* DDSAsyncTask * *task,* long *timeout*)

This function adds a asynchronous task to the asynchronous thread.

Parameters

query	The DDS query condition that is used to take the request. Cannot be NULL.
task	The asynchronos task created and associated with a request. Cannot be NULL.
timeout	The timeout used for this request.

Returns

A 0 value is returned if function works successfully. In any other case, -1 is returned.

6.42.3.2 FASTRPC_DllAPI eprosima::rpc::transport::Endpoint* eprosima::rpc::transport::dds::ProxyTransport::createProcedureEndpoint (const char * *name,* const char * *writertypename,* const char * *readertypename,* bool *eprosima_types,* Transport::Create_data *create_data,* Transport::Copy_data *copy_data,* Transport::Destroy_data *destroy_data,* Transport::ProcessFunc *processFunc,* int *dataSize*) [virtual]

This function creates a new proxy procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.

TODO Actualizar

Parameters

name	The name associated with this proxy procedure endpoint. It cannot be NULL.
writertypename	The type name of the topic that the procedure endpoint uses in the datawriter. It cannot be NULL.
readertypename	The type name of the topic that the procedure endpoint uses in the datareader. It cannot be NULL.

initialize_data	Pointer to the function to initialize DataReader received data
copy_data	Pointer to the function used to copy the data when it is received.
finalize_data	Pointer to the function to finalize DataReader received data
ProcessFunc	Pointer to the function invoked when a message is received from the server
dataSize	Size of the DataReader data structure

Returns

0 if the function works. -1 in other case. TODO

Implements eprosima::rpc::transport::dds::Transport.

6.42.3.3 FASTRPC_DllAPI void eprosima::rpc::transport::dds::ProxyTransport::deleteAssociatedAsyncTasks (ProxyProcedureEndpoint * *pe*)

This function deletes all the asynchronous tasks associated with the ProxyProcedureEndpoint endpoint.

Parameters

pe	Pointer to the ProxyProcedureEndpoint. It cannot be NULL.

6.42.3.4 FASTRPC_DllAPI const char* eprosima::rpc::transport::dds::ProxyTransport::getRemoteServiceName ( ) const

This function returns the DDS service name.

Returns

DDS service name.

6.42.3.5 FASTRPC_DllAPI long eprosima::rpc::transport::dds::ProxyTransport::getTimeout ( )

This function gets the timeout value.

Returns

Timeout value.

6.42.3.6 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::ProxyTransport::setTransport (DDS_DomainParticipantQos & *participantQos*, DDSDomainParticipant * *participant*) [protected], [pure virtual]

This abstract function sets the QoS to use a specific transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use a specific transport.

Implements eprosima::rpc::transport::dds::Transport.

Implemented in eprosima::rpc::transport::dds::UDPProxyTransport.

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

- includetmp/rpcdds/transports/dds/ProxyTransport.h

6.43 eprosima::rpc::transport::ProxyTransport Class Reference

This interface is the base of all classes that implement a transport that can be used by the proxy.

`#include <ProxyTransport.h>`

Inheritance diagram for eprosima::rpc::transport::ProxyTransport:

Collaboration diagram for eprosima::rpc::transport::ProxyTransport:

Public Member Functions

- ProxyTransport ()
  Default constructor.
- virtual ~ProxyTransport ()
  Default destructor.
- virtual const char * getType () const =0
  This function returns the type of the transport. This function has to be implemented by the child classes.
- TransportBehaviour getBehaviour () const
  This function returns the behaviour of the transport.
- virtual bool connect ()=0
  Abstract method. It must start a connection with the server.
- virtual bool send (const void *buffer, const size_t bufferSize)=0
  Abstract method. It must send a request to the server.
- virtual int receive (void *buffer, const size_t bufferSize, size_t &dataToRead)=0
  Abstract method. It must receive a reply from the server.

6.43.1 Detailed Description

This interface is the base of all classes that implement a transport that can be used by the proxy.

6.43.2 Member Function Documentation

6.43.2.1 virtual bool eprosima::rpc::transport::ProxyTransport::connect ( ) [pure virtual]

Abstract method. It must start a connection with the server.

Returns

true if the operation is successful, false otherwise.

6.43.2.2 TransportBehaviour eprosima::rpc::transport::ProxyTransport::getBehaviour ( ) const [inline], [virtual]

This function returns the behaviour of the transport.

Returns

The behaviour of the transport.

Implements eprosima::rpc::transport::Transport.

6.43.2.3 virtual int eprosima::rpc::transport::ProxyTransport::receive (void * *buffer*, const size_t *bufferSize*, size_t & *dataToRead*) [pure virtual]

Abstract method. It must receive a reply from the server.

Parameters

buffer	Buffer that will contain the HTTP message.
bufferSize	Size of the buffer.
dataToRead	Number of bytes received.

Returns

-1 if the operation fails.

6.43.2.4 virtual bool eprosima::rpc::transport::ProxyTransport::send (const void * *buffer*, const size_t *bufferSize*) [pure virtual]

Abstract method. It must send a request to the server.

Parameters

buffer	Buffer containing the request
bufferSize	Buffer size

Returns

true if the operation is successful, false otherwise.

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

- includetmp/rpcdds/transports/ProxyTransport.h

6.44 eprosima::rpc::protocol::dds::ReplyHeader Class Reference

Header information used in all generated reply topics.

`#include <MessageHeader.h>`

Public Member Functions

- ReplyHeader ()
  Default constructor.
- ReplyHeader (const ReplyHeader &header)
  Copy constructor.
- ReplyHeader (ReplyHeader &&header)
  Copy constructor.
- ~ReplyHeader ()
  Destructor.
- ReplyHeader & operator= (const ReplyHeader &header)
  Copy assignment.
- ReplyHeader & operator= (ReplyHeader &&header)
  Copy assignment.
- void request_id (const SampleIdentity_t &_request_id)
  This function sets the client identifier.
- void request_id (SampleIdentity_t &&_request_id)
  This function sets the client identifier.
- const SampleIdentity_t & request_id () const

This function returns the client identifier.

- SampleIdentity_t & request_id ()

  This function returns the client identifier.

- void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes the ReplyHeader object using CDR serialization.

- void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes the ReplyHeader object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment)

  This function returns the maximum serialized size of a ReplyHeader object depending on the buffer alignment.

6.44.1 Detailed Description

Header information used in all generated reply topics.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 eprosima::rpc::protocol::dds::ReplyHeader::ReplyHeader (const ReplyHeader & *header*)

Copy constructor.

Parameters

header	ReplyHeader object to be copied.

6.44.2.2 eprosima::rpc::protocol::dds::ReplyHeader::ReplyHeader (ReplyHeader && *header*)

Copy constructor.

Parameters

header	ReplyHeader object to be copied.

6.44.3 Member Function Documentation

6.44.3.1 void eprosima::rpc::protocol::dds::ReplyHeader::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes the ReplyHeader object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.44.3.2 static size_t eprosima::rpc::protocol::dds::ReplyHeader::getMaxCdrSerializedSize (size_t *current_alignment*) [static]

This function returns the maximum serialized size of a ReplyHeader object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.44.3.3 ReplyHeader& eprosima::rpc::protocol::dds::ReplyHeader::operator= (const ReplyHeader & *header*)

Copy assignment.

Parameters

header	ReplyHeader object to be copied.

6.44.3.4 ReplyHeader& eprosima::rpc::protocol::dds::ReplyHeader::operator= (ReplyHeader && *header*)

Copy assignment.

Parameters

header	ReplyHeader object to be copied.

6.44.3.5 void eprosima::rpc::protocol::dds::ReplyHeader::request_id (const SampleIdentity_t & *_request_id*) [inline]

This function sets the client identifier.

Parameters

_clientId	Client identifier

6.44.3.6 void eprosima::rpc::protocol::dds::ReplyHeader::request_id (SampleIdentity_t && *_request_id*) [inline]

This function sets the client identifier.

Parameters

_clientId	Client identifier

6.44.3.7 const SampleIdentity_t& eprosima::rpc::protocol::dds::ReplyHeader::request_id ( ) const [inline]

This function returns the client identifier.

Returns

Client identifier

6.44.3.8 SampleIdentity_t& eprosima::rpc::protocol::dds::ReplyHeader::request_id ( ) [inline]

This function returns the client identifier.

Returns

Client identifier

6.44.3.9 void eprosima::rpc::protocol::dds::ReplyHeader::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes the ReplyHeader object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- includetmp/rpcdds/protocols/dds/MessageHeader.h

6.45 eprosima::rpc::protocol::dds::ReplyHeaderPlugin Class Reference

This class offers the functions needed by DDS middleware to use the class ReplyHeaderPlugin.

`#include <MessageHeaderPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()

  This function returns the TypeCode.

6.45.1 Detailed Description

This class offers the functions needed by DDS middleware to use the class ReplyHeaderPlugin.

6.45.2 Member Function Documentation

6.45.2.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::ReplyHeaderPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.46 eprosima::rpc::protocol::dds::RequestHeader Class Reference

Header information used in all generated request topics.

`#include <MessageHeader.h>`

Public Member Functions

- RequestHeader ()

  Default constructor.
- RequestHeader (const RequestHeader &header)

  Copy constructor.
- RequestHeader (RequestHeader &&header)

  Copy constructor.
- ~RequestHeader ()

  Destructor.
- RequestHeader & operator= (const RequestHeader &header)

  Copy assignment.
- RequestHeader & operator= (RequestHeader &&header)

  Copy assignment.
- void request_id (const SampleIdentity_t &_request_id)

  This function sets the client identifier.
- void request_id (SampleIdentity_t &&_request_id)

  This function sets the client identifier.
- const SampleIdentity_t & request_id () const

  This function returns the client identifier.
- SampleIdentity_t & request_id ()

  This function returns the client identifier.
- void remote_service_name (const char *_remote_service_name)

  This function sets the server service name.
- const char * remote_service_name () const

  This function returns the server service name.
- void instance_name (const char *_instance_name)

  This function sets the server service name.
- const char * instance_name () const

  This function returns the server service name.
- void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes the RequestHeader object using CDR serialization.
- void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes the RequestHeader object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment)

  This function returns the maximum serialized size of a RequestHeader object depending on the buffer alignment.

6.46.1 Detailed Description

Header information used in all generated request topics.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 eprosima::rpc::protocol::dds::RequestHeader::RequestHeader (const RequestHeader & *header*)

Copy constructor.

Parameters

header	RequestHeader object to be copied.

6.46.2.2 eprosima::rpc::protocol::dds::RequestHeader::RequestHeader (RequestHeader && *header*)

Copy constructor.

Parameters

header	RequestHeader object to be copied.

6.46.3 Member Function Documentation

6.46.3.1 void eprosima::rpc::protocol::dds::RequestHeader::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes the RequestHeader object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.46.3.2 static size_t eprosima::rpc::protocol::dds::RequestHeader::getMaxCdrSerializedSize (size_t *current_alignment*) [static]

This function returns the maximum serialized size of a RequestHeader object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.46.3.3 void eprosima::rpc::protocol::dds::RequestHeader::instance_name (const char * *_instance_name*) [inline]

This function sets the server service name.

Parameters

_remoteServiceName	Server service name.

6.46.3.4 const char* eprosima::rpc::protocol::dds::RequestHeader::instance_name ( ) const [inline]

This function returns the server service name.

Returns

Server service name.

6.46.3.5 RequestHeader& eprosima::rpc::protocol::dds::RequestHeader::operator= (const RequestHeader & *header*)

Copy assignment.

Parameters

header	RequestHeader object to be copied.

6.46.3.6 RequestHeader& eprosima::rpc::protocol::dds::RequestHeader::operator= (RequestHeader && *header*)

Copy assignment.

Parameters

header	RequestHeader object to be copied.

6.46.3.7 void eprosima::rpc::protocol::dds::RequestHeader::remote_service_name (const char * *_remote_service_name*) [inline]

This function sets the server service name.

Parameters

_remoteService-Name	Server service name.

6.46.3.8 const char* eprosima::rpc::protocol::dds::RequestHeader::remote_service_name ( ) const [inline]

This function returns the server service name.

Returns

Server service name.

6.46.3.9 void eprosima::rpc::protocol::dds::RequestHeader::request_id (const SampleIdentity_t & *_request_id*) [inline]

This function sets the client identifier.

Parameters

_clientId	Client identifier

6.46.3.10 void eprosima::rpc::protocol::dds::RequestHeader::request_id (SampleIdentity_t && *_request_id*) [inline]

This function sets the client identifier.

Parameters

_clientId	Client identifier

6.46.3.11 const SampleIdentity_t& eprosima::rpc::protocol::dds::RequestHeader::request_id ( ) const [inline]

This function returns the client identifier.

Returns

Client identifier

6.46.3.12 SampleIdentity_t& eprosima::rpc::protocol::dds::RequestHeader::request_id ( ) [inline]

This function returns the client identifier.

Returns

Client identifier

6.46.3.13 void eprosima::rpc::protocol::dds::RequestHeader::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes the RequestHeader object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- includetmp/rpcdds/protocols/dds/MessageHeader.h

6.47 eprosima::rpc::protocol::dds::RequestHeaderPlugin Class Reference

This class offers the functions needed by DDS middleware to use the class RequestHeaderPlugin.

`#include <MessageHeaderPlugin.h>`

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()
  This function returns the TypeCode.

6.47.1 Detailed Description

This class offers the functions needed by DDS middleware to use the class RequestHeaderPlugin.

6.47.2 Member Function Documentation

6.47.2.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::RequestHeaderPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.48 eprosima::rpc::protocol::dds::SampleIdentity_t Class Reference

This class is used to identify clients.

`#include <MessageHeader.h>`

Public Member Functions

- SampleIdentity_t ()

  Default constructor.
- SampleIdentity_t (const SampleIdentity_t &id)

  Copy constructor.
- SampleIdentity_t (SampleIdentity_t &&id)

  Copy constructor.
- ~SampleIdentity_t ()

  Destructor.
- SampleIdentity_t & operator= (const SampleIdentity_t &id)

  Copy assignment.
- SampleIdentity_t & operator= (SampleIdentity_t &&id)

  Copy assignment.
- void **guid** (const GUID_t &_guid)
- void **guid** (GUID_t &&_guid)
- const GUID_t & guid () const

  This function returns the client identifier.
- GUID_t & guid ()

  This function returns the client identifier.
- void sequence_number (int64_t _sequence_number)

  This function sets the fourth value of the client identifier.
- int64_t sequence_number () const

  This function returns the fourth value of the client identifier.
- int64_t & sequence_number ()

  This function returns the fourth value of the client identifier.
- void serialize (eprosima::fastcdr::Cdr &cdr) const

  This function serializes the SampleIdentity_t object using CDR serialization.
- void deserialize (eprosima::fastcdr::Cdr &cdr)

  This function deserializes the SampleIdentity_t object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment)

  This function returns the maximum serialized size of a SampleIdentity_t object depending on the buffer alignment.

6.48.1 Detailed Description

This class is used to identify clients.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 eprosima::rpc::protocol::dds::SampleIdentity_t::SampleIdentity_t (const SampleIdentity_t & *id*)

Copy constructor.

Parameters

id	SampleIdentity_t object to be copied.

6.48.2.2 eprosima::rpc::protocol::dds::SampleIdentity_t::SampleIdentity_t (SampleIdentity_t && *id*)

Copy constructor.

Parameters

id	SampleIdentity_t object to be copied.

6.48.3 Member Function Documentation

6.48.3.1 void eprosima::rpc::protocol::dds::SampleIdentity_t::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes the SampleIdentity_t object using CDR serialization.

Parameters

cdr	CDR serialization object.

6.48.3.2 static size_t eprosima::rpc::protocol::dds::SampleIdentity_t::getMaxCdrSerializedSize (size_t *current_alignment*) [static]

This function returns the maximum serialized size of a SampleIdentity_t object depending on the buffer alignment.

Parameters

current_alignment	Buffer alignment.

Returns

Maximum serialized size.

6.48.3.3 const GUID_t& eprosima::rpc::protocol::dds::SampleIdentity_t::guid ( ) const [inline]

This function returns the client identifier.

Returns

Client identifier

6.48.3.4 GUID_t& eprosima::rpc::protocol::dds::SampleIdentity_t::guid ( ) [inline]

This function returns the client identifier.

Returns

Client identifier

6.48.3.5 SampleIdentity_t& eprosima::rpc::protocol::dds::SampleIdentity_t::operator= (const SampleIdentity_t & *id*)

Copy assignment.

Parameters

id	SampleIdentity_t object to be copied.

6.48.3.6 SampleIdentity_t& eprosima::rpc::protocol::dds::SampleIdentity_t::operator= (SampleIdentity_t && *id*)

Copy assignment.

Parameters

id	SampleIdentity_t object to be copied.

6.48.3.7 void eprosima::rpc::protocol::dds::SampleIdentity_t::sequence_number (int64_t *_sequence_number*) [inline]

This function sets the fourth value of the client identifier.

Parameters

_value_4	Fourth value of the client identifier.

6.48.3.8 int64_t eprosima::rpc::protocol::dds::SampleIdentity_t::sequence_number ( ) const [inline]

This function returns the fourth value of the client identifier.

Returns

Fourth value of the client identifier.

6.48.3.9 int64_t& eprosima::rpc::protocol::dds::SampleIdentity_t::sequence_number ( ) [inline]

This function returns the fourth value of the client identifier.

Returns

Fourth value of the client identifier.

6.48.3.10 void eprosima::rpc::protocol::dds::SampleIdentity_t::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes the SampleIdentity_t object using CDR serialization.

Parameters

cdr	CDR serialization object.

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

- includetmp/rpcdds/protocols/dds/MessageHeader.h

6.49 eprosima::rpc::protocol::dds::SampleIdentity_tPlugin Class Reference

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()

  This function returns the TypeCode.

6.49.1 Member Function Documentation

6.49.1.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::SampleIdentity_tPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.50 eprosima::rpc::server::Server Class Reference

This class implements the common functionalities that any server has.

`#include <Server.h>`

Inheritance diagram for eprosima::rpc::server::Server:

Public Member Functions

- void serve ()

  This function makes the server starts listening requests.
  Exceptions

eprosima::rpc::exception::InitializeException	*This exception is thrown when the initialization fails for any reason.*

- void stop ()

  This function closes the server's communications.

Static Public Member Functions

- static void process (Server &server, void *data, eprosima::rpc::transport::Endpoint &endpoint)

  This callback is invoked by the ServerStrategy. It processes a request.

Protected Member Functions

- Server (eprosima::rpc::strategy::ServerStrategy &strategy, eprosima::rpc::transport::ServerTransport &transport, eprosima::rpc::protocol::Protocol &protocol)

  A constructor. The associated domain participant is created.
- virtual ~Server ()

  The default destructor.

6.50.1 Detailed Description

This class implements the common functionalities that any server has.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 eprosima::rpc::server::Server::Server (eprosima::rpc::strategy::ServerStrategy & *strategy,* eprosima::rpc::transport::ServerTransport & *transport,* eprosima::rpc::protocol::Protocol & *protocol*) [protected]

A constructor. The associated domain participant is created.

Parameters

serviceName	The service's name that proxies will use to connect with the server.
strategy	The strategy used by the server to execute new requests. This class doesn't delete this object in its destructor. It cannot be NULL.
transport	The transport that will use the server. This class doesn't delete this object in its destructor. If the pointer is NULL, then a default UDPTransport will be used.
domainId	The domain id's value that the server proxy will set in the domain participant.

Exceptions

InitializeException	This exception is thrown when the initialization was wrong.

6.50.3 Member Function Documentation

6.50.3.1 static void eprosima::rpc::server::Server::process (Server & *server,* void * *data,* eprosima::rpc::transport::Endpoint & *endpoint*) [static]

This callback is invoked by the ServerStrategy. It processes a request.

Parameters

server	The invoked server.
data	The request data.
endpoint	The request endpoint.

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

- includetmp/rpcdds/server/Server.h

6.51 eprosima::rpc::exception::ServerInternalException Class Reference

This class is thrown as an exception when there is an error in the server side.

`#include <ServerInternalException.h>`

Inheritance diagram for eprosima::rpc::exception::ServerInternalException:

Collaboration diagram for eprosima::rpc::exception::ServerInternalException:

Public Member Functions

- FASTRPC_DllAPI ServerInternalException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI ServerInternalException (const ServerInternalException &ex)

  Default copy constructor.
- FASTRPC_DllAPI ServerInternalException (ServerInternalException &&ex)

  Default move constructor.
- FASTRPC_DllAPI ServerInternalException & operator= (const ServerInternalException &ex)

  Assigment operation.
- FASTRPC_DllAPI ServerInternalException & operator= (ServerInternalException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~ServerInternalException () throw ()

  Default constructor.

- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.51.1 Detailed Description

This class is thrown as an exception when there is an error in the server side.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 FASTRPC_DllAPI eprosima::rpc::exception::ServerInternalException::ServerInternalException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.51.2.2 FASTRPC_DllAPI eprosima::rpc::exception::ServerInternalException::ServerInternalException (const ServerInternalException & *ex*)

Default copy constructor.

Parameters

ex	ServerInternalException that will be copied.

6.51.2.3 FASTRPC_DllAPI eprosima::rpc::exception::ServerInternalException::ServerInternalException (ServerInternalException && *ex*)

Default move constructor.

Parameters

ex	ServerInternalException that will be moved.

6.51.3 Member Function Documentation

6.51.3.1 FASTRPC_DllAPI ServerInternalException& eprosima::rpc::exception::ServerInternalException::operator= (const ServerInternalException & *ex*)

Assigment operation.

Parameters

ex	ServerInternalException that will be copied.

6.51.3.2 FASTRPC_DllAPI ServerInternalException& eprosima::rpc::exception::ServerInternalException::operator= (ServerInternalException && *ex*)

Assigment operation.

Parameters

ex	ServerInternalException that will be moved.

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

- includetmp/rpcdds/exceptions/ServerInternalException.h

6.52 eprosima::rpc::exception::ServerNotFoundException Class Reference

This class is thrown as an exception when the server is not found.

`#include <ServerNotFoundException.h>`

Inheritance diagram for eprosima::rpc::exception::ServerNotFoundException:

Collaboration diagram for eprosima::rpc::exception::ServerNotFoundException:

Public Member Functions

- FASTRPC_DllAPI ServerNotFoundException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI ServerNotFoundException (const ServerNotFoundException &ex)

  Default copy constructor.
- FASTRPC_DllAPI ServerNotFoundException (ServerNotFoundException &&ex)

  Default move constructor.
- FASTRPC_DllAPI ServerNotFoundException & operator= (const ServerNotFoundException &ex)

  Assigment operation.
- FASTRPC_DllAPI ServerNotFoundException & operator= (ServerNotFoundException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~ServerNotFoundException () throw ()

  Default constructor.
- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.52.1 Detailed Description

This class is thrown as an exception when the server is not found.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 FASTRPC_DllAPI eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.52.2.2 FASTRPC_DllAPI eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (const ServerNotFoundException & *ex*)

Default copy constructor.

Parameters

ex	ServerNotFoundException that will be copied.

6.52.2.3 FASTRPC_DllAPI eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (ServerNotFoundException && *ex*)

Default move constructor.

Parameters

ex	ServerNotFoundException that will be moved.

6.52.3 Member Function Documentation

6.52.3.1 FASTRPC_DllAPI ServerNotFoundException& eprosima::rpc::exception::ServerNotFoundException::operator= (const ServerNotFoundException & *ex*)

Assigment operation.

Parameters

ex	ServerNotFoundException that will be copied.

6.52.3.2 FASTRPC_DllAPI ServerNotFoundException& eprosima::rpc::exception::ServerNotFoundException::operator= (ServerNotFoundException && *ex*)

Assigment operation.

Parameters

ex	ServerNotFoundException that will be moved.

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

- includetmp/rpcdds/exceptions/ServerNotFoundException.h

6.53 eprosima::rpc::transport::dds::ServerProcedureEndpoint Class Reference

This class represents a remote endpoint used by a proxy. Also this class encapsulate the DDS datawriter and the DDS datareader.

`#include <ServerProcedureEndpoint.h>`

Inheritance diagram for eprosima::rpc::transport::dds::ServerProcedureEndpoint:

Collaboration diagram for eprosima::rpc::transport::dds::ServerProcedureEndpoint:

Public Member Functions

- FASTRPC_DllAPI ServerProcedureEndpoint (ServerTransport &transport)

  Default constructor.
- virtual FASTRPC_DllAPI ~ServerProcedureEndpoint ()

  Default destructor.
- FASTRPC_DllAPI int initialize (const char *name, const char *writertypename, const char *readertypename, Transport::Create_data create_data, Transport::Destroy_data destroy_data, Transport::ProcessFunc, int dataSize)

  Initializes the endpoint.
- FASTRPC_DllAPI int start (const char *const &serviceName, const char *const &instanceName)

  This method creates the DDS entities needed to run this DDS Endpoint.
- FASTRPC_DllAPI void stop ()

  This method deletes the DDS entities needed to run this DDS Endpoint.
- FASTRPC_DllAPI Transport::ProcessFunc getProcessFunc ()

  Gets the callback used to processes a request.
- FASTRPC_DllAPI int sendReply (void *data)

  Sends the reply.
- virtual FASTRPC_DllAPI void on_data_available (DDSDataReader *reader)

  DDS callback.
- virtual FASTRPC_DllAPI void on_requested_deadline_missed (DDSDataReader *reader, const DDS_RequestedDeadlineMissedStatus &status)

  DDS callback.
- virtual FASTRPC_DllAPI void on_requested_incompatible_qos (DDSDataReader *reader, const DDS_RequestedIncompatibleQosStatus &status)

  DDS callback.
- virtual FASTRPC_DllAPI void on_sample_rejected (DDSDataReader *reader, const DDS_SampleRejectedStatus &status)

  DDS callback.
- virtual FASTRPC_DllAPI void on_liveliness_changed (DDSDataReader *reader, const DDS_LivelinessChangedStatus &status)

  DDS callback.
- virtual FASTRPC_DllAPI void on_sample_lost (DDSDataReader *reader, const DDS_SampleLostStatus &status)

  DDS callback.
- virtual FASTRPC_DllAPI void on_subscription_matched (DDSDataReader *reader, const DDS_SubscriptionMatchedStatus &status)

  DDS callback.

Additional Inherited Members

6.53.1 Detailed Description

This class represents a remote endpoint used by a proxy. Also this class encapsulate the DDS datawriter and the DDS datareader.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::ServerProcedureEndpoint::ServerProcedureEndpoint (ServerTransport & *transport*)

Default constructor.

Parameters

Transport	that creates the proxy procedure endpoint. It cannot be NULL.

6.53.3 Member Function Documentation

6.53.3.1 FASTRPC_DllAPI Transport::ProcessFunc eprosima::rpc::transport::dds::ServerProcedureEndpoint::getProcessFunc ( ) [inline]

Gets the callback used to processes a request.

Returns

Function callback used to processes a request.

6.53.3.2 FASTRPC_DllAPI int eprosima::rpc::transport::dds::ServerProcedureEndpoint::initialize (const char * *name,* const char * *writertypename,* const char * *readertypename,* Transport::Create_data *create_data,* Transport::Destroy_data *destroy_data,* Transport::ProcessFunc , int *dataSize*)

Initializes the endpoint.

TODO Actualizar

Parameters

name	The name associated with this procedure endpoint. It cannot be NULL.
writertypename	The type name of the topic that the procedure endpoint uses in the datawriter. It cannot be NULL.
readertypename	The type name of the topic that the procedure endpoint uses in the datareader. It cannot be NULL.
initialize_data	Pointer to the function to initialize DataReader received data
finalize_data	Pointer to the function to finalize DataReader received data
ProcessFunc	Pointer to the function invoked when a message is received from the server
dataSize	Size of the DataReader data structure

6.53.3.3 FASTRPC_DllAPI int eprosima::rpc::transport::dds::ServerProcedureEndpoint::sendReply (void * *data*)

Sends the reply.

Parameters

serviceName	Name of the service.

6.53.3.4 FASTRPC_DllAPI int eprosima::rpc::transport::dds::ServerProcedureEndpoint::start (const char *const & *serviceName,* const char *const & *instanceName*)

This method creates the DDS entities needed to run this DDS Endpoint.

Parameters

serviceName	Name of the service.

6.53.3.5 FASTRPC_DllAPI void eprosima::rpc::transport::dds::ServerProcedureEndpoint::stop ( )

This method deletes the DDS entities needed to run this DDS Endpoint.

Parameters

serviceName	Name of the service.

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

- includetmp/rpcdds/transports/dds/components/ServerProcedureEndpoint.h

6.54 eprosima::rpc::strategy::ServerStrategy Class Reference

This class is the base of all classes that implement a server strategy. that could be used by the server.

`#include <ServerStrategy.h>`

Inheritance diagram for eprosima::rpc::strategy::ServerStrategy:

Public Member Functions

- ServerStrategy ()

  Default constructor.
- virtual ~ServerStrategy ()

  Default destructor.
- virtual ServerStrategyImpl * getImpl ()=0

  Gets the implementation of the strategy using Boost library.

6.54.1 Detailed Description

This class is the base of all classes that implement a server strategy. that could be used by the server.

6.54.2 Member Function Documentation

6.54.2.1 virtual ServerStrategyImpl* eprosima::rpc::strategy::ServerStrategy::getImpl ( ) [pure virtual]

Gets the implementation of the strategy using Boost library.

Returns

Implementation of the strategy.

Implemented in eprosima::rpc::strategy::ThreadPoolStrategy, eprosima::rpc::strategy::SingleThreadStrategy, and eprosima::rpc::strategy::ThreadPerRequestStrategy.

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

- includetmp/rpcdds/strategies/ServerStrategy.h

6.55 eprosima::rpc::strategy::ServerStrategyImpl Class Reference

This class is the base of all classes that implement a server strategy. that could be used by the server.

`#include <ServerStrategyImpl.h>`

Public Member Functions

- ServerStrategyImpl ()

  Default constructor.
- virtual ~ServerStrategyImpl ()

  Default destructor.
- virtual void schedule (boost::function< void()> callback)=0

  This function schedules an incoming request. This function has to be implemented by the derived classes.

6.55.1 Detailed Description

This class is the base of all classes that implement a server strategy. that could be used by the server.

6.55.2 Member Function Documentation

6.55.2.1 virtual void eprosima::rpc::strategy::ServerStrategyImpl::schedule (boost::function< void()> *callback*) [pure virtual]

This function schedules an incoming request. This function has to be implemented by the derived classes.

Parameters

callback	The Server's method to invoke when a request arrives.

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

- includetmp/rpcdds/strategies/ServerStrategyImpl.h

6.56 eprosima::rpc::exception::ServerTimeoutException Class Reference

This class is thrown as an exception when the remote procedure call exceeds the maximum time.

`#include <ServerTimeoutException.h>`

Inheritance diagram for eprosima::rpc::exception::ServerTimeoutException:

Collaboration diagram for eprosima::rpc::exception::ServerTimeoutException:

Public Member Functions

- FASTRPC_DllAPI ServerTimeoutException (const std::string &message)

  Default constructor.
- FASTRPC_DllAPI ServerTimeoutException (const ServerTimeoutException &ex)

  Default copy constructor.
- FASTRPC_DllAPI ServerTimeoutException (ServerTimeoutException &&ex)

  Default move constructor.
- FASTRPC_DllAPI ServerTimeoutException & operator= (const ServerTimeoutException &ex)

  Assigment operation.
- FASTRPC_DllAPI ServerTimeoutException & operator= (ServerTimeoutException &&ex)

  Assigment operation.
- virtual FASTRPC_DllAPI ~ServerTimeoutException () throw ()

  Default constructor.

- virtual FASTRPC_DllAPI void raise () const

  This function throws the object as an exception.

Additional Inherited Members

6.56.1 Detailed Description

This class is thrown as an exception when the remote procedure call exceeds the maximum time.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 FASTRPC_DllAPI eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (const std::string & *message*) [inline]

Default constructor.

Parameters

message	An error message. This message is copied.

6.56.2.2 FASTRPC_DllAPI eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (const **ServerTimeoutException** & *ex*)

Default copy constructor.

Parameters

ex	ServerTimeoutException that will be copied.

6.56.2.3 FASTRPC_DllAPI eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (**ServerTimeoutException** && *ex*)

Default move constructor.

Parameters

ex	ServerTimeoutException that will be moved.

6.56.3 Member Function Documentation

6.56.3.1 FASTRPC_DllAPI **ServerTimeoutException**& eprosima::rpc::exception::ServerTimeoutException::operator= (const **ServerTimeoutException** & *ex*)

Assigment operation.

Parameters

ex	ServerTimeoutException that will be copied.

6.56.3.2 FASTRPC_DllAPI **ServerTimeoutException**& eprosima::rpc::exception::ServerTimeoutException::operator= (**ServerTimeoutException** && *ex*)

Assigment operation.

Parameters

ex	ServerTimeoutException that will be moved.

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

- includetmp/rpcdds/exceptions/ServerTimeoutException.h

6.57 eprosima::rpc::transport::ServerTransport Class Reference

This interface is the base of all classes that implement a transport that can be used by the server.

`#include <ServerTransport.h>`

Inheritance diagram for eprosima::rpc::transport::ServerTransport:

Collaboration diagram for eprosima::rpc::transport::ServerTransport:

Public Member Functions

- ServerTransport ()

  Default constructor.
- virtual ~ServerTransport ()

  Default destructor.
- void setStrategy (eprosima::rpc::strategy::ServerStrategy &strategy)

  Sets the threading strategy.
- void linkProtocol (eprosima::rpc::protocol::Protocol &protocol)

  Sets the communication protocol.
- eprosima::rpc::protocol::Protocol & getLinkedProtocol ()

  Gets the communication protocol.
- eprosima::rpc::strategy::ServerStrategy & getStrategy () const

  Gets the threading strategy.
- ServerTransport_Callback getCallback () const

  Gets the callback that will proccess the requests.
- void setCallback (ServerTransport_Callback callback)

  Gets the callback that will proccess the requests.
- TransportBehaviour getBehaviour () const

  This function returns the behaviour of the transport.
- virtual const char * getType () const =0

  This function returns the type of the transport. This function has to be implemented by the child classes.
- virtual void run ()=0

  This method runs the TCP server needed for the HTTP connections.
- virtual void stop ()=0

  This method stops the TCP server needed for the HTTP connections.
- virtual void sendReply (void *data, size_t dataLength, Endpoint *endpoint)=0

  This function is used to send a reply to a proxy.
- virtual int receive (char *buffer, size_t bufferLength, size_t &dataToRead, Endpoint *endpoint)=0

  This function is used to send a reply to a proxy.

6.57.1 Detailed Description

This interface is the base of all classes that implement a transport that can be used by the server.

6.57.2 Member Function Documentation

6.57.2.1 TransportBehaviour eprosima::rpc::transport::ServerTransport::getBehaviour ( ) const [inline], [virtual]

This function returns the behaviour of the transport.

Returns

The behaviour of the transport.

Implements eprosima::rpc::transport::Transport.

6.57.2.2 ServerTransport_Callback eprosima::rpc::transport::ServerTransport::getCallback ( ) const [inline]

Gets the callback that will proccess the requests.

Returns

Callback that will proccess the requests.

6.57.2.3 eprosima::rpc::protocol::Protocol& eprosima::rpc::transport::ServerTransport::getLinkedProtocol ( ) [inline]

Gets the communication protocol.

Returns

Communication protocol.

6.57.2.4 eprosima::rpc::strategy::ServerStrategy& eprosima::rpc::transport::ServerTransport::getStrategy ( ) const [inline]

Gets the threading strategy.

Returns

Threading strategy.

6.57.2.5 void eprosima::rpc::transport::ServerTransport::linkProtocol (eprosima::rpc::protocol::Protocol & *protocol*) [inline]

Sets the communication protocol.

Parameters

protocol	Communication protocol.

6.57.2.6 virtual int eprosima::rpc::transport::ServerTransport::receive (char * *buffer*, size_t *bufferLength*, size_t & *dataToRead*, Endpoint * *endpoint*) [pure virtual]

This function is used to send a reply to a proxy.

Parameters

buffer	Buffer to allocate the received data
bufferLength	Size of the buffer
dataToRead	Size of the data to read
endpoint	Endpoint to receive the data from

Implemented in eprosima::rpc::transport::dds::ServerTransport.

6.57.2.7 virtual void eprosima::rpc::transport::ServerTransport::sendReply (void * *data*, size_t *dataLength*, Endpoint * *endpoint*) [pure virtual]

This function is used to send a reply to a proxy.

Parameters

data	Response to send.
dataLength	Length of the data to send.
endpoint	Targeg entpoint to send the data to.

Implemented in eprosima::rpc::transport::dds::ServerTransport.

6.57.2.8 void eprosima::rpc::transport::ServerTransport::setCallback (ServerTransport_Callback *callback*) [inline]

Gets the callback that will proccess the requests.

Parameters

Callback	Callback that will proccess the requests.

6.57.2.9 void eprosima::rpc::transport::ServerTransport::setStrategy (eprosima::rpc::strategy::ServerStrategy & *strategy*) [inline]

Sets the threading strategy.

Parameters

strategy	Threading strategy.

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

- includetmp/rpcdds/transports/ServerTransport.h

6.58 eprosima::rpc::transport::dds::ServerTransport Class Reference

This class is the base of all classes that implement a transport using DDS. This transport can be used by the servers.

`#include <ServerTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::ServerTransport:

Collaboration diagram for eprosima::rpc::transport::dds::ServerTransport:

Public Member Functions

- virtual FASTRPC_DllAPI ~ServerTransport ()

  Default destructor.

- virtual FASTRPC_DllAPI const char * getType () const

  This function returns the type of the transport. This function has to be implemented by the child classes.

- FASTRPC_DllAPI eprosima::rpc::transport::Endpoint * createProcedureEndpoint (const char *name, const char *writertypename, const char *readertypename, bool eprosima_types, Transport::Create_data create_data, Transport::Copy_data copy_data, Transport::Destroy_data destroy_data, Transport::ProcessFunc processFunc, int dataSize)

  This function creates a new proxy procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.

- FASTRPC_DllAPI void process (ServerProcedureEndpoint *endpoint, void *data)

  This method is invoked once for each incoming request.

- FASTRPC_DllAPI void run ()

  This method starts all the DDS Datawriters and Datareaders.

- FASTRPC_DllAPI void stop ()

  This function does not apply to DDS transport.

- void sendReply (void *data, size_t dataLength, Endpoint *endpoint)

  This function is used to send a reply to a proxy.

- FASTRPC_DllAPI int receive (char *buffer, size_t bufferLength, size_t &dataToRead, Endpoint *endpoint)

  This function does not apply to DDS transport.

Protected Member Functions

- virtual FASTRPC_DllAPI int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant *participant)=0

  This abstract function sets the QoS of DDS to use a specific transport.

- FASTRPC_DllAPI ServerTransport (const char *const &serviceName, const char *const &instanceName, int domainId=0)

  Default constructor.

Additional Inherited Members

6.58.1 Detailed Description

This class is the base of all classes that implement a transport using DDS. This transport can be used by the servers.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::ServerTransport::ServerTransport (const char *const & *serviceName*, const char *const & *instanceName*, int *domainId* = `0`) [protected]

Default constructor.

Parameters

domainId	Optional parameter that specifies the domain identifier that will be used in DDS.

6.58.3 Member Function Documentation

6.58.3.1 FASTRPC_DllAPI eprosima::rpc::transport::Endpoint* eprosima::rpc::transport::dds::ServerTransport::createProcedureEndpoint (const char * *name*, const char * *writertypename*, const char * *readertypename*, bool *eprosima_types*, Transport::Create_data *create_data*, Transport::Copy_data *copy_data*, Transport::Destroy_data *destroy_data*, Transport::ProcessFunc *processFunc*, int *dataSize*) [virtual]

This function creates a new proxy procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.

TODO Actualizar

Parameters

name	The name associated with this proxy procedure endpoint. It cannot be NULL.
writertypename	The type name of the topic that the procedure endpoint uses in the datawriter. It cannot be NULL.
readertypename	The type name of the topic that the procedure endpoint uses in the datareader. It cannot be NULL.
initialize_data	Pointer to the function to initialize DataReader received data
copy_data	Pointer to the function used to copy the data when it is received.
finalize_data	Pointer to the function to finalize DataReader received data
ProcessFunc	Pointer to the function invoked when a message is received from the server
dataSize	Size of the DataReader data structure

Returns

0 if the function successfully works, -1 in other case TODO

Implements eprosima::rpc::transport::dds::Transport.

6.58.3.2 FASTRPC_DllAPI void eprosima::rpc::transport::dds::ServerTransport::process (ServerProcedureEndpoint * *endpoint*, void * *data*)

This method is invoked once for each incoming request.

Parameters

data	The request data.
endpoint	The request endpoint.

6.58.3.3 void eprosima::rpc::transport::dds::ServerTransport::sendReply (void * *data*, size_t *dataLength*, Endpoint * *endpoint*) [virtual]

This function is used to send a reply to a proxy.

Parameters

data	Data to send.
dataLength	Length of the data to send.
endpoint	Endpoint meant to send the data.

Implements eprosima::rpc::transport::ServerTransport.

6.58.3.4 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::ServerTransport::setTransport (DDS_DomainParticipantQos & *participantQos*, DDSDomainParticipant * *participant*) [protected], [pure virtual]

This abstract function sets the QoS of DDS to use a specific transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use a specific transport.

Implements eprosima::rpc::transport::dds::Transport.

Implemented in eprosima::rpc::transport::dds::TCPServerTransport, and eprosima::rpc::transport::dds::UDPServerTransport.

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

- includetmp/rpcdds/transports/dds/ServerTransport.h

6.59 eprosima::rpc::strategy::SingleThreadStrategy Class Reference

This class implements the sigle thread strategy. The server uses a reception thread to execute all the requests.

`#include <SingleThreadStrategy.h>`

Inheritance diagram for eprosima::rpc::strategy::SingleThreadStrategy:

Collaboration diagram for eprosima::rpc::strategy::SingleThreadStrategy:

Public Member Functions

- SingleThreadStrategy ()

  Default constructor.
- virtual ~SingleThreadStrategy ()

  Default destructor.
- ServerStrategyImpl * getImpl ()

  Gets the implementation of the strategy using Boost library.

6.59.1 Detailed Description

This class implements the sigle thread strategy. The server uses a reception thread to execute all the requests.

6.59.2 Member Function Documentation

6.59.2.1 ServerStrategyImpl* eprosima::rpc::strategy::SingleThreadStrategy::getImpl ( ) [virtual]

Gets the implementation of the strategy using Boost library.

Returns

Implementation of the strategy

Implements eprosima::rpc::strategy::ServerStrategy.

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

- includetmp/rpcdds/strategies/SingleThreadStrategy.h

6.60 eprosima::rpc::exception::SystemException Class Reference

This abstract class is used to create internal FASTRPC exceptions.

`#include <SystemException.h>`

Inheritance diagram for eprosima::rpc::exception::SystemException:

Collaboration diagram for eprosima::rpc::exception::SystemException:

Public Member Functions

- virtual FASTRPC_DllAPI ~SystemException () throw ()

  Default destructor.
- FASTRPC_DllAPI const int32_t & minor () const

  This function returns the number associated with the system exception.
- FASTRPC_DllAPI void minor (const int32_t &minor)

  This function sets the number that will be associated with the system exception.
- virtual FASTRPC_DllAPI void raise () const =0

  This function throws the object as an exception.
- virtual FASTRPC_DllAPI const char * what () const throw ()

  This function returns the error message.

Protected Member Functions

- FASTRPC_DllAPI SystemException (const char *const &message)

  Default constructor.
- FASTRPC_DllAPI SystemException (const SystemException &ex)

  Default copy constructor.
- FASTRPC_DllAPI SystemException (SystemException &&ex)

  Default move constructor.
- FASTRPC_DllAPI SystemException (const char *const &message, int32_t minor)

  Constructor.
- FASTRPC_DllAPI SystemException & operator= (const SystemException &ex)

  Assigment operation.
- FASTRPC_DllAPI SystemException & operator= (SystemException &&ex)

  Assigment operation.

6.60.1 Detailed Description

This abstract class is used to create internal FASTRPC exceptions.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 FASTRPC_DllAPI eprosima::rpc::exception::SystemException::SystemException (const char *const & *message*) [explicit], [protected]

Default constructor.

Parameters

message	An error message. This message is copied.

6.60.2.2 FASTRPC_DllAPI eprosima::rpc::exception::SystemException::SystemException (const SystemException & *ex*) [protected]

Default copy constructor.

Parameters

ex	SystemException that will be copied.

6.60.2.3 FASTRPC_DllAPI eprosima::rpc::exception::SystemException::SystemException (SystemException && *ex*) [protected]

Default move constructor.

Parameters

ex	SystemException that will be moved.

6.60.2.4 FASTRPC_DllAPI eprosima::rpc::exception::SystemException::SystemException (const char *const & *message,* int32_t *minor*) [explicit], [protected]

Constructor.

Parameters

message	An error message. This message is copied.
minor	The number that will be associated with the system exception.

6.60.3 Member Function Documentation

6.60.3.1 FASTRPC_DllAPI const int32_t& eprosima::rpc::exception::SystemException::minor ( ) const

This function returns the number associated with the system exception.

Returns

The number associated with the system exception.

6.60.3.2 FASTRPC_DllAPI void eprosima::rpc::exception::SystemException::minor (const int32_t & *minor*)

This function sets the number that will be associated with the system exception.

Parameters

minor	The number that will be associated with the system exception.

6.60.3.3 FASTRPC_DllAPI SystemException& eprosima::rpc::exception::SystemException::operator= (const SystemException & *ex*) [protected]

Assigment operation.

Parameters

ex	SystemException that will be copied.

6.60.3.4 FASTRPC_DllAPI SystemException& eprosima::rpc::exception::SystemException::operator= (SystemException && *ex*) [protected]

Assigment operation.

Parameters

ex	SystemException that will be moved.

6.60.3.5 virtual FASTRPC_DllAPI const char* eprosima::rpc::exception::SystemException::what ( ) const throw) [virtual]

This function returns the error message.

Returns

The error message.

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

- includetmp/rpcdds/exceptions/SystemException.h

6.61 eprosima::rpc::protocol::dds::SystemExceptionCodePlugin Class Reference

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()
  This function returns the TypeCode.

6.61.1 Member Function Documentation

6.61.1.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::SystemExceptionCodePlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.62 eprosima::rpc::transport::dds::TCPProxyTransport Class Reference

This class implements a transport using DDS over TCPv4. This transport can only be used by a server proxy.

`#include <TCPProxyTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::TCPProxyTransport:

Collaboration diagram for eprosima::rpc::transport::dds::TCPProxyTransport:

Public Member Functions

- FASTRPC_DllAPI TCPProxyTransport (const char ∗const &to_connect, const char ∗const &remoteServiceName, const char ∗const &instanceName, int domainId=0, long timeout=10000L)

  Default constructor for the proxies.
- virtual FASTRPC_DllAPI ∼TCPProxyTransport ()

  Default destructor.
- virtual FASTRPC_DllAPI int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)

  This function sets the DDS' QoS to use the TCPv4 transport.

Additional Inherited Members

6.62.1 Detailed Description

This class implements a transport using DDS over TCPv4. This transport can only be used by a server proxy.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::TCPProxyTransport::TCPProxyTransport (const char ∗const & *to_connect*, const char ∗const & *remoteServiceName*, const char ∗const & *instanceName*, int *domainId* = `0`, long *timeout* = `10000L`)

Default constructor for the proxies.

Parameters

to_connect	Public address and port where the server can be found by the proxy. By example: "218.18.-3.133:7600"
remoteService-Name	Name of the remote service
domainId	Optional parameter that specifies the domain identifier to be used in DDS.
timeout	The time in milliseconds to wait for the reply.

6.62.3 Member Function Documentation

6.62.3.1 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::TCPProxyTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *participant*) `[virtual]`

This function sets the DDS' QoS to use the TCPv4 transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use TCPv4 transport.

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

- includetmp/rpcdds/transports/dds/TCPProxyTransport.h

6.63 eprosima::rpc::transport::dds::TCPServerTransport Class Reference

This class implements a transport using DDS over TCPv4. This transport can only be used by a server.

`#include <TCPServerTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::TCPServerTransport:

Collaboration diagram for eprosima::rpc::transport::dds::TCPServerTransport:

Public Member Functions

- FASTRPC_DllAPI TCPServerTransport (const char ∗const &public_address, const char ∗const &server_bind_port, const char ∗const &serviceName, const char ∗const &instanceName, int domainId=0)

  Default constructor for servers.
- virtual FASTRPC_DllAPI ∼TCPServerTransport ()

  Default destructor.
- virtual FASTRPC_DllAPI int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant ∗participant)

  This function sets the QoS to use the TCPv4 transport.

Additional Inherited Members

6.63.1 Detailed Description

This class implements a transport using DDS over TCPv4. This transport can only be used by a server.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::TCPServerTransport::TCPServerTransport (const char ∗const & *public_address*, const char ∗const & *server_bind_port*, const char ∗const & *serviceName*, const char ∗const & *instanceName*, int *domainId* = `0`)

Default constructor for servers.

Parameters

public_address	Public address and port of the server. The server should be accesible in this address. The user has to configure his router for this purpose. For example: "218.18.3.133:7600"
server_bind_port	Port used by the server in his machine. This port will be used in the router for port forwarding between the public port and this port.
serviceName	Name of the remote service
domainId	Optional parameter that specifies the domain identifier to be used in DDS.

6.63.3 Member Function Documentation

6.63.3.1 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::TCPServerTransport::setTransport (DDS_DomainParticipantQos & *participantQos*, DDSDomainParticipant ∗ *participant*) `[virtual]`

This function sets the QoS to use the TCPv4 transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use TCPv4 transport.

Implements eprosima::rpc::transport::dds::ServerTransport.

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

- includetmp/rpcdds/transports/dds/TCPServerTransport.h

6.64 eprosima::rpc::strategy::ThreadPerRequestStrategy Class Reference

This class implements the thread per request strategy. The server creates a new thread for every new incoming request.

`#include <ThreadPerRequestStrategy.h>`

Inheritance diagram for eprosima::rpc::strategy::ThreadPerRequestStrategy:

Collaboration diagram for eprosima::rpc::strategy::ThreadPerRequestStrategy:

Public Member Functions

- ThreadPerRequestStrategy ()

  Default constructor.
- virtual ~ThreadPerRequestStrategy ()

  Default destructor.
- ServerStrategyImpl * getImpl ()

  Gets the implementation of the strategy using Boost library.

6.64.1 Detailed Description

This class implements the thread per request strategy. The server creates a new thread for every new incoming request.

6.64.2 Member Function Documentation

6.64.2.1 ServerStrategyImpl* eprosima::rpc::strategy::ThreadPerRequestStrategy::getImpl ( ) [virtual]

Gets the implementation of the strategy using Boost library.

Returns

Strategy implementation.

Implements eprosima::rpc::strategy::ServerStrategy.

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

- includetmp/rpcdds/strategies/ThreadPerRequestStrategy.h

6.65 eprosima::rpc::strategy::ThreadPoolStrategy Class Reference

This class implements a thread pool strategy. The server schedules the incoming requests in a free thread of the thread pool.

`#include <ThreadPoolStrategy.h>`

Inheritance diagram for eprosima::rpc::strategy::ThreadPoolStrategy:

Collaboration diagram for eprosima::rpc::strategy::ThreadPoolStrategy:

Public Member Functions

- ThreadPoolStrategy (unsigned int threadCount=FASTRPC_MIN_THREADS_DEFAULT)

Default constructor.

- ~ThreadPoolStrategy ()

  Default destructor.

- ServerStrategyImpl * getImpl ()

  Gets the implementation of the strategy using Boost library.

6.65.1 Detailed Description

This class implements a thread pool strategy. The server schedules the incoming requests in a free thread of the thread pool.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 eprosima::rpc::strategy::ThreadPoolStrategy::ThreadPoolStrategy (unsigned int *threadCount* = FASTRPC_MIN_THREADS_DEFAULT)

Default constructor.

Parameters

threadCount	Number of threads the thread pool will manage. Default value: 5.

6.65.3 Member Function Documentation

6.65.3.1 ServerStrategyImpl* eprosima::rpc::strategy::ThreadPoolStrategy::getImpl ( ) [virtual]

Gets the implementation of the strategy using Boost library.

Returns

Implementation of the strategy.

Implements eprosima::rpc::strategy::ServerStrategy.

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

- includetmp/rpcdds/strategies/ThreadPoolStrategy.h

6.66 eprosima::rpc::transport::dds::Transport Class Reference

This class is the base of all classes that implement a transport using DDS. This transport could be used by both proxies and servers.

`#include <Transport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::Transport:

Public Types

- typedef void *(* **Create_data**)(void)
- typedef void(* **Copy_data**)(void *dst, void *src)
- typedef void(* **Destroy_data**)(void *data)
- typedef void(* **ProcessFunc**)(eprosima::rpc::protocol::Protocol &, void *, eprosima::rpc::transport::Endpoint *)

Public Member Functions

- virtual ~Transport ()

  Default destructor.
- void initialize ()

  Initializes all the DDS elements: creates the topic, the participant, the publisher and the subscriber.
- DDSDomainParticipant * getParticipant () const

  Gets the domain participant.
- DDSPublisher * getPublisher () const

  Gets the publisher.
- DDSSubscriber * getSubscriber () const

  Gets the subscriber.
- virtual eprosima::rpc::transport::Endpoint * createProcedureEndpoint (const char *name, const char *writertypename, const char *readertypename, bool eprosima_types, Create_data create_data, Copy_data copy_data, Destroy_data destroy_data, ProcessFunc processFunc, int dataSize)=0

  This function creates a new procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.

Protected Member Functions

- virtual int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant *participant)=0

  This abstract function sets the QoS to use a specific transport.
- Transport (int domainId=0)

  Default constructor.

6.66.1 Detailed Description

This class is the base of all classes that implement a transport using DDS. This transport could be used by both proxies and servers.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 eprosima::rpc::transport::dds::Transport::Transport (int *domainId* = 0) [protected]

Default constructor.

Parameters

domainId	Optional parameter that specifies the domain identifier that will be used in DDS.

6.66.3 Member Function Documentation

6.66.3.1 virtual eprosima::rpc::transport::Endpoint* eprosima::rpc::transport::dds::Transport::createProcedureEndpoint (const char * *name*, const char * *writertypename*, const char * *readertypename*, bool *eprosima_types*, Create_data *create_data*, Copy_data *copy_data*, Destroy_data *destroy_data*, ProcessFunc *processFunc*, int *dataSize*) [pure virtual]

This function creates a new procedure endpoint. This proxy procedure endpoint manages the DDS datawriter and the DDS datareader.

TODO Actualizar

Parameters

name	The name associated with this proxy procedure endpoint. It cannot be NULL.
writertypename	The type name of the topic that the procedure endpoint uses in the datawriter. It cannot be NULL.
readertypename	The type name of the topic that the procedure endpoint uses in the datareader. It cannot be NULL.
initialize_data	Pointer to the function to initialize DataReader received data
copy_data	Pointer to the function used to copy the data when it is received.
finalize_data	Pointer to the function to finalize DataReader received data
ProcessFunc	Pointer to the function invoked when a message is received from the server
dataSize	Size of the DataReader data structure

Returns

0 if the function ends successfully, -1 otherwise. TODO

Implemented in eprosima::rpc::transport::dds::ProxyTransport, and eprosima::rpc::transport::dds::ServerTransport.

6.66.3.2 DDSDomainParticipant* eprosima::rpc::transport::dds::Transport::getParticipant ( ) const [inline]

Gets the domain participant.

Returns

DDS domain participant.

6.66.3.3 DDSPublisher* eprosima::rpc::transport::dds::Transport::getPublisher ( ) const [inline]

Gets the publisher.

Returns

DDS publisher.

6.66.3.4 DDSSubscriber* eprosima::rpc::transport::dds::Transport::getSubscriber ( ) const [inline]

Gets the subscriber.

Returns

DDS subscriber.

6.66.3.5 virtual int eprosima::rpc::transport::dds::Transport::setTransport (DDS_DomainParticipantQos & *participantQos*, DDSDomainParticipant * *participant*) [protected], [pure virtual]

This abstract function sets the QoS to use a specific transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.

participant	The domain participant that will be set to use a specific transport.

Implemented in eprosima::rpc::transport::dds::ProxyTransport, eprosima::rpc::transport::dds::ServerTransport, eprosima::rpc::transport::dds::UDPProxyTransport, eprosima::rpc::transport::dds::TCPServerTransport, and eprosima::rpc::transport::dds::UDPServerTransport.

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

- includetmp/rpcdds/transports/dds/Transport.h

6.67 eprosima::rpc::transport::Transport Class Reference

This class is the base of all classes that implement a transport that could be used by the proxy or the server.

`#include <Transport.h>`

Inheritance diagram for eprosima::rpc::transport::Transport:

Public Member Functions

- Transport ()
  Default constructor.
- virtual ~Transport ()
  Default destructor.
- virtual const char * getType () const =0
  This function returns the type of the transport. This function has to be implemented by the child classes.
- virtual TransportBehaviour getBehaviour () const =0

6.67.1 Detailed Description

This class is the base of all classes that implement a transport that could be used by the proxy or the server.

6.67.2 Member Function Documentation

6.67.2.1 virtual TransportBehaviour eprosima::rpc::transport::Transport::getBehaviour ( ) const [pure virtual]

2brief This function returns the behaviour of the transport.

Returns

The behaviour of the transport.

Implemented in eprosima::rpc::transport::ServerTransport, and eprosima::rpc::transport::ProxyTransport.

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

- includetmp/rpcdds/transports/Transport.h

6.68 eprosima::rpc::transport::dds::UDPProxyTransport Class Reference

This class implements a transport using DDS over UDPv4. This transport only can be used by a server's proxy.

`#include <UDPProxyTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::UDPProxyTransport:

Collaboration diagram for eprosima::rpc::transport::dds::UDPProxyTransport:

Public Member Functions

- FASTRPC_DllAPI UDPProxyTransport (const char ∗const &remoteServiceName, const char ∗const &instanceName, int domainId=0, long timeout=10000L)

  Default constructor for server's proxies.
- FASTRPC_DllAPI UDPProxyTransport (const char ∗const &to_connect, const char ∗const &remoteServiceName, const char ∗const &instanceName, int domainId=0, long timeout=10000L)

  Constructor for server's proxies.
- virtual FASTRPC_DllAPI ∼UDPProxyTransport ()

  Default destructor.
- virtual FASTRPC_DllAPI int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant ∗participant)

  This function sets the QoS of DDS to use the UDPv4 transport.

Additional Inherited Members

6.68.1 Detailed Description

This class implements a transport using DDS over UDPv4. This transport only can be used by a server's proxy.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::UDPProxyTransport::UDPProxyTransport (const char ∗const & *remoteServiceName,* const char ∗const & *instanceName,* int *domainId* = `0`, long *timeout* = `10000L`)

Default constructor for server's proxies.

Parameters

remoteServiceName	Name of the service
domainId	Optional parameter that specifies the domain identifier to be used in DDS.
timeout	The time in milliseconds to wait for the reply.

6.68.2.2 FASTRPC_DllAPI eprosima::rpc::transport::dds::UDPProxyTransport::UDPProxyTransport (const char ∗const & *to_connect,* const char ∗const & *remoteServiceName,* const char ∗const & *instanceName,* int *domainId* = `0`, long *timeout* = `10000L`)

Constructor for server's proxies.

Parameters

to_connect	IP address where the server can be found by the proxy. For example: "192.168.1.3"
remoteServiceName	Name of the service
domainId	Optional parameter that specifies the domain identifier to be used in DDS.
timeout	The time in milliseconds to wait for the reply.

6.68.3 Member Function Documentation

6.68.3.1 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::UDPProxyTransport::setTransport (DDS_DomainParticipantQos & *participantQos,* DDSDomainParticipant ∗ *participant*) `[virtual]`

This function sets the QoS of DDS to use the UDPv4 transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use UDPv4 transport.

Implements eprosima::rpc::transport::dds::ProxyTransport.

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

- includetmp/rpcdds/transports/dds/UDPProxyTransport.h

6.69 eprosima::rpc::transport::dds::UDPServerTransport Class Reference

This class implements transport using DDS over UDPv4. This transport can only be used by a server.

`#include <UDPServerTransport.h>`

Inheritance diagram for eprosima::rpc::transport::dds::UDPServerTransport:

Collaboration diagram for eprosima::rpc::transport::dds::UDPServerTransport:

Public Member Functions

- FASTRPC_DllAPI UDPServerTransport (const char ∗const &serviceName, const char ∗const &instanceName, int domainId=0)

  Default constructor for servers.
- virtual FASTRPC_DllAPI ∼UDPServerTransport ()

  Default destructor.
- virtual FASTRPC_DllAPI int setTransport (DDS_DomainParticipantQos &participantQos, DDSDomainParticipant ∗participant)

  This function sets the DDS' QoS to use the UDPv4 transport.

Additional Inherited Members

6.69.1 Detailed Description

This class implements transport using DDS over UDPv4. This transport can only be used by a server.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 FASTRPC_DllAPI eprosima::rpc::transport::dds::UDPServerTransport::UDPServerTransport (const char ∗const & *serviceName,* const char ∗const & *instanceName,* int *domainId* = `0`)

Default constructor for servers.

Parameters

remoteServiceName	Name of the service
domainId	Optional parameter that specifies the domain identifier that will be used in DDS.

6.69.3 Member Function Documentation

6.69.3.1 virtual FASTRPC_DllAPI int eprosima::rpc::transport::dds::UDPServerTransport::setTransport (DDS_DomainParticipantQos & *participantQos*, DDSDomainParticipant * *participant*) [virtual]

This function sets the DDS' QoS to use the UDPv4 transport.

Parameters

participantQos	Reference to the DDS domain participant QoS.
participant	The domain participant that will be set to use UDPv4 transport.

Implements eprosima::rpc::transport::dds::ServerTransport.

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

- includetmp/rpcdds/transports/dds/UDPServerTransport.h

6.70 eprosima::rpc::protocol::dds::UnknownExceptionPlugin Class Reference

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()

  This function returns the TypeCode.

6.70.1 Member Function Documentation

6.70.1.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::UnknownExceptionPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.71 eprosima::rpc::protocol::dds::UnknownOperationPlugin Class Reference

Static Public Member Functions

- static DDS_TypeCode * get_typecode ()

  This function returns the TypeCode.

6.71.1 Member Function Documentation

6.71.1.1 static DDS_TypeCode* eprosima::rpc::protocol::dds::UnknownOperationPlugin::get_typecode ( ) [static]

This function returns the TypeCode.

Returns

The TypeCode.

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

- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

6.72 eprosima::rpc::exception::UserException Class Reference

This abstract class is used to create user exceptions.

`#include <UserException.h>`

Inheritance diagram for eprosima::rpc::exception::UserException:

Collaboration diagram for eprosima::rpc::exception::UserException:

Public Member Functions

- virtual ~UserException () throw ()

  Default destructor.
- virtual void raise () const =0

  This function throws the object as exception.

Protected Member Functions

- UserException ()

  Default constructor.
- UserException (const UserException &ex)

  Default copy constructor.
- UserException (UserException &&ex)

  Default move constructor.
- UserException & operator= (const UserException &ex)

  Assigment operation.
- UserException & operator= (UserException &&ex)

  Assigment operation.

6.72.1 Detailed Description

This abstract class is used to create user exceptions.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 eprosima::rpc::exception::UserException::UserException (const UserException & *ex*) [protected]

Default copy constructor.

Parameters

ex	UserException that will be copied.

6.72.2.2 eprosima::rpc::exception::UserException::UserException (UserException && *ex*) [protected]

Default move constructor.

Parameters

ex	UserException that will be moved.

6.72.3 Member Function Documentation

6.72.3.1 UserException& eprosima::rpc::exception::UserException::operator= (const UserException & *ex*) [protected]

Assigment operation.

Parameters

ex	UserException that will be copied.

6.72.3.2 UserException& eprosima::rpc::exception::UserException::operator= (UserException && *ex*) [protected]

Assigment operation.

Parameters

ex	UserException that will be moved.

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

- includetmp/rpcdds/exceptions/UserException.h

Index