eProsima RPCDDS C++ API

Version 0.3.2

Generated by Doxygen 1.6.1

Fri Aug 29 13:29:43 2014

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

Namespace Index

3.1 Namespace List

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

Chapter 4

Class Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

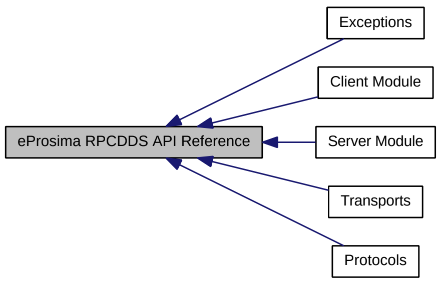
Chapter 6

Module Documentation

6.1 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.

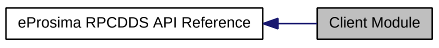
6.1.1 Detailed Description

eProsima RPC over DDS internal API grouped in modules.

6.2 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.

6.2.1 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.

6.3 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:

eProsima RPCDDS API Reference
Server Module
Strategies

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.

Modules

- Strategies

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

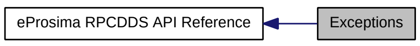
6.3.1 Detailed Description

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.

6.4 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.

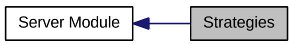
6.4.1 Detailed Description

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*.

6.5 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.

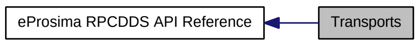
6.5.1 Detailed Description

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

6.6 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.

- class eprosima::rpc::transport::ProxyTransport

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

- class eprosima::rpc::transport::ServerTransport

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

- class eprosima::rpc::transport::Transport

  This class is the base of all classes that implement a transport that could be used by the proxy or the 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.

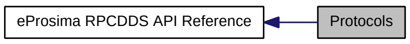
6.6.1 Detailed Description

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

6.7 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::Identification

  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::IdentificationPlugin

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

- 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.

6.7.1 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.

6.8 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

- 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 eprosima::rpc::protocol::FooDDSProtocol

  Protocol base class for the specific application.

- 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_FooProcedureRequest

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

- class FooDDS::Foo_FooProcedureReply

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

- class FooDDS::FooRequest_union

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

- class FooDDS::FooRequest

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

- class FooDDS::FooReply_union

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

- class FooDDS::FooReply

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

Namespaces

- namespace FooDDS::Foo

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

6.8.1 Detailed Description

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

Chapter 7

Namespace Documentation

7.1 FooDDS::Foo Namespace Reference

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

7.1.1 Detailed Description

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

Chapter 8

Class Documentation

8.1 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>
```

Inherited by eprosima::rpc::transport::dds::DDSAsyncTask.

Protected Member Functions

- AsyncTask ()

  Default constructor.

- virtual ∼AsyncTask ()

  Destructor.

8.1.1 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

8.2 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 (DDS::QueryCondition ∗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.

8.2.1 Detailed Description

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

8.2.2 Member Function Documentation

8.2.2.1 int eprosima::rpc::transport::dds::AsyncThread::addTask (DDS::QueryCondition ∗ *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

8.2.2.2 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.

8.2.2.3 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:

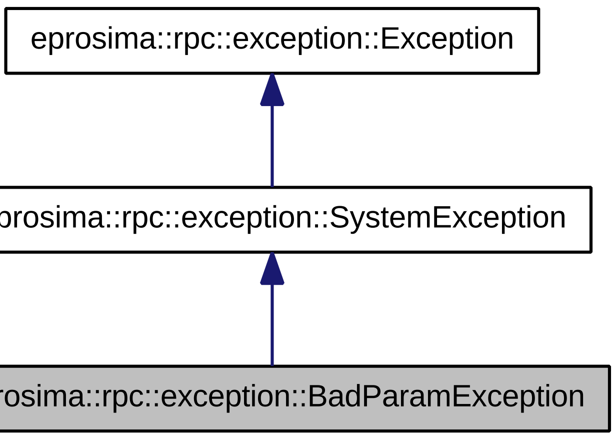
- includetmp/rpcdds/transports/dds/AsyncThread.h

8.3 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>
```

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::BadParamException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::BadParamException

Public Member Functions

- BadParamException (const std::string &message)

  Default constructor.

- BadParamException (std::string &&message)

  Default constructor.

- BadParamException (const BadParamException &ex)

  Default copy constructor.

- BadParamException (BadParamException &&ex)

  Default move constructor.

- BadParamException & operator= (const BadParamException &ex)

  Assigment operation.

- BadParamException & operator= (BadParamException &&ex)

  Assigment operation.

- virtual ~BadParamException () throw ()

Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.3.1 Detailed Description

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

8.3.2 Constructor & Destructor Documentation

8.3.2.1 eprosima::rpc::exception::BadParamException::BadParamException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.3.2.2 eprosima::rpc::exception::BadParamException::BadParamException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.3.2.3 eprosima::rpc::exception::BadParamException::BadParamException (const BadParamException & *ex*)

Default copy constructor.

Parameters:

ex BadParamException that will be copied.

8.3.2.4 eprosima::rpc::exception::BadParamException::BadParamException (BadParamException && *ex*)

Default move constructor.

Parameters:

ex BadParamException that will be moved.

8.3.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex BadParamException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

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

Assigment operation.

Parameters:

ex BadParamException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

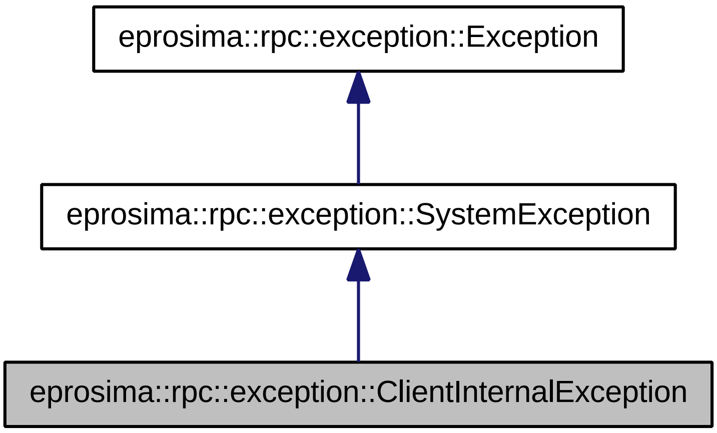
- includetmp/rpcdds/exceptions/BadParamException.h

8.4 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>
```

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::ClientInternalException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::ClientInternalException

Public Member Functions

- ClientInternalException (const std::string &message)

  Default constructor.

- ClientInternalException (std::string &&message)

  Default constructor.

- ClientInternalException (const ClientInternalException &ex)

  Default copy constructor.

- ClientInternalException (ClientInternalException &&ex)

  Default move constructor.

- ClientInternalException & operator= (const ClientInternalException &ex)

  Assigment operation.

- ClientInternalException & operator= (ClientInternalException &&ex)

  Assigment operation.

- virtual ~ClientInternalException () throw ()

  Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.4.1 Detailed Description

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

8.4.2 Constructor & Destructor Documentation

8.4.2.1 eprosima::rpc::exception::ClientInternalException::ClientInternalException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.4.2.2 eprosima::rpc::exception::ClientInternalException::ClientInternalException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.4.2.3 eprosima::rpc::exception::ClientInternalException::ClientInternalException (const ClientInternalException & *ex*)

Default copy constructor.

Parameters:

ex ClientInternalException that will be copied.

8.4.2.4 eprosima::rpc::exception::ClientInternalException::ClientInternalException (ClientInternalException && *ex*)

Default move constructor.

Parameters:

ex ClientInternalException that will be moved.

8.4.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex ClientInternalException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

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

Assigment operation.

Parameters:

ex ClientInternalException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

- includetmp/rpcdds/exceptions/ClientInternalException.h

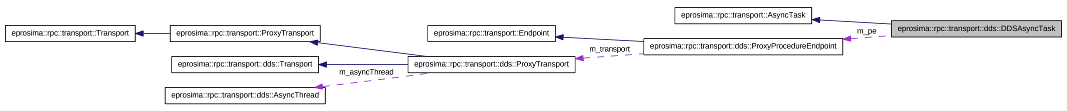
8.5 eprosima::rpc::transport::dds::DDSAsyncTask Class Reference

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

```
#include <DDSAsyncTask.h>
```

Inherits eprosima::rpc::transport::AsyncTask.

Inherited by FooDDS::Foo_FooProcedureTask.Collaboration diagram for eprosima::rpc::transport::dds::DDSAsyncTask:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::Endpoint
eprosima::rpc::transport::AsyncTask
eprosima::rpc::transport::dds::DDSAsyncTask
m_pe
eprosima::rpc::transport::dds::ProxyProcedureEndpoint
m_transport
eprosima::rpc::transport::dds::Transport
m_asyncThread
eprosima::rpc::transport::dds::ProxyTransport
eprosima::rpc::transport::dds::AsyncThread

Public Member Functions

- DDSAsyncTask ()

  default constructor

- virtual ∼DDSAsyncTask ()

  default destructor

- void execute (DDS::QueryCondition ∗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.

8.5.1 Detailed Description

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

8.5.2 Member Function Documentation

8.5.2.1 void eprosima::rpc::transport::dds::DDSAsyncTask::execute (DDS::QueryCondition ∗ *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.

8.5.2.2 ProxyProcedureEndpoint∗ eprosima::rpc::transport::dds::DDSAsyncTask::getProcedureEndpoint ()

Gets the procedure endpoint.

Returns:

Procedure endpoint with the DDS datawriter and datareader

8.5.2.3 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.

8.5.2.4 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.

8.5.2.5 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

8.6 eprosima::rpc::transport::Endpoint Class Reference

This class represents an endpoint.

`#include <Endpoint.h>`

Inherited by eprosima::rpc::transport::dds::ProxyProcedureEndpoint, and eprosima::rpc::transport::dds::ServerProcedureEndpoint.

Protected Member Functions

- Endpoint ()

  Default constructor.

- virtual ∼Endpoint ()

  Default destructor.

8.6.1 Detailed Description

This class represents an endpoint.

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

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

8.7 eprosima::rpc::exception::Exception Class Reference

This abstract class is used to create exceptions.

`#include <Exception.h>`

Inherited by eprosima::rpc::exception::SystemException, and eprosima::rpc::exception::UserException.

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.

8.7.1 Detailed Description

This abstract class is used to create exceptions.

8.7.2 Constructor & Destructor Documentation

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

Default copy constructor.

Parameters:

ex Exception that will be copied.

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

Default move constructor.

Parameters:

ex Exception that will be moved.

8.7.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex Exception that will be moved.

Reimplemented in eprosima::rpc::exception::BadParamException, eprosima::rpc::exception::ClientInternalException, eprosima::rpc::exception::IncompatibleException, eprosima::rpc::exception::InitializeException, eprosima::rpc::exception::ServerInternalException, eprosima::rpc::exception::ServerNotFoundException, eprosima::rpc::exception::ServerTimeoutException, eprosima::rpc::exception::SystemException, and eprosima::rpc::exception::UserException.

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

Assigment operation.

Parameters:

ex Exception that will be copied.

Reimplemented in eprosima::rpc::exception::BadParamException, eprosima::rpc::exception::ClientInternalException, eprosima::rpc::exception::IncompatibleException, eprosima::rpc::exception::InitializeException, eprosima::rpc::exception::ServerInternalException, eprosima::rpc::exception::ServerNotFoundException, eprosima::rpc::exception::ServerTimeoutException, eprosima::rpc::exception::SystemException, and eprosima::rpc::exception::UserException.

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

- includetmp/rpcdds/exceptions/Exception.h

8.8 FooDDS::Foo Interface Reference

Public Member Functions

- void **FooProcedure** ()

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

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

8.9 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.

8.9.1 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.

8.9.2 Member Function Documentation

8.9.2.1 virtual void FooDDS::Foo_-FooProcedureCallbackHandler::FooProcedure () `[pure virtual]`

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

8.9.2.2 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

8.10 FooDDS::Foo_FooProcedureReply Class Reference

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

`#include <FooDDSTopics.h>`

Public Member Functions

- Foo_FooProcedureReply ()

  Default constructor.

- ∼Foo_FooProcedureReply ()

  Destructor.

- Foo_FooProcedureReply (const Foo_FooProcedureReply &x)

  Copy constructor.

- Foo_FooProcedureReply (Foo_FooProcedureReply &&x)

  Move constructor.

- Foo_FooProcedureReply & operator= (const Foo_FooProcedureReply &x)

  Copy assignment.

- Foo_FooProcedureReply & operator= (Foo_FooProcedureReply &&x)

  Copy assignment.

- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function serializes an object using CDR serialization.

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.10.1 Detailed Description

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

8.10.2 Constructor & Destructor Documentation

8.10.2.1 FooDDS::Foo_FooProcedureReply::Foo_FooProcedureReply (const Foo_FooProcedureReply & *x*)

Copy constructor.

Parameters:

x Reference to the object Foo_FooProcedureReply that will be copied.

8.10.2.2 FooDDS::Foo_FooProcedureReply::Foo_FooProcedureReply (Foo_FooProcedureReply && *x*)

Move constructor.

Parameters:

x Reference to the object Foo_FooProcedureReply that will be copied.

8.10.3 Member Function Documentation

8.10.3.1 void FooDDS::Foo_FooProcedureReply::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.10.3.2 size_t FooDDS::Foo_FooProcedureReply::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.

8.10.3.3 size_t FooDDS::Foo_FooProcedureReply::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.

8.10.3.4 FooDDS::Foo_FooProcedureReply & FooDDS::Foo_-FooProcedureReply::operator= (FooDDS::Foo_FooProcedureReply && *x*)

Copy assignment.

Parameters:

x Reference to the object Foo_FooProcedure that will be copied.

8.10.3.5 FooDDS::Foo_FooProcedureReply & FooDDS::Foo_-FooProcedureReply::operator= (const Foo_FooProcedureReply & *x*)

Copy assignment.

Parameters:

x Reference to the object Foo_FooProcedure that will be copied.

8.10.3.6 void FooDDS::Foo_FooProcedureReply::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

8.11 FooDDS::Foo_FooProcedureRequest Class Reference

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

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

Public Member Functions

- Foo_FooProcedureRequest ()

  Default constructor.
- ~Foo_FooProcedureRequest ()

  Destructor.
- **Foo_FooProcedureRequest** (const Foo_FooProcedureRequest &x)
- Foo_FooProcedureRequest (Foo_FooProcedureRequest &&x)

  Move constructor.
- Foo_FooProcedureRequest & operator= (const Foo_FooProcedureRequest &x)

  Copy assignment.
- Foo_FooProcedureRequest & operator= (Foo_FooProcedureRequest &&x)

  Copy assignment.
- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.11.1 Detailed Description

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

8.11.2 Constructor & Destructor Documentation

8.11.2.1 FooDDS::Foo_FooProcedureRequest::Foo_FooProcedureRequest (Foo_FooProcedureRequest && *x*)

Move constructor.

Parameters:

x Reference to the object Foo_FooProcedureRequest that will be copied.

8.11.3 Member Function Documentation

8.11.3.1 void FooDDS::Foo_FooProcedureRequest::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.11.3.2 size_t FooDDS::Foo_FooProcedureRequest::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.

8.11.3.3 size_t FooDDS::Foo_FooProcedureRequest::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.

8.11.3.4 FooDDS::Foo_FooProcedureRequest & FooDDS::Foo_FooProcedureRequest::operator= (FooDDS::Foo_FooProcedureRequest && *x*)

Copy assignment.

Parameters:

x Reference to the object Foo_FooProcedure that will be copied.

8.11.3.5 FooDDS::Foo_FooProcedureRequest & FooDDS::Foo_-FooProcedureRequest::operator= (const Foo_FooProcedureRequest & *x*)

Copy assignment.

Parameters:

x Reference to the object Foo_FooProcedure that will be copied.

8.11.3.6 void FooDDS::Foo_FooProcedureRequest::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:

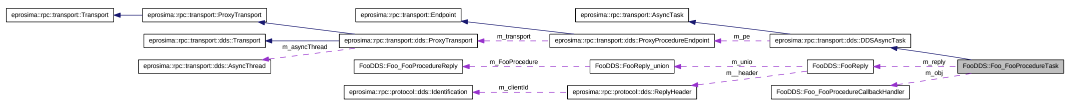
- utils/doxygen/examples/dds/FooDDSTopics.h
- utils/doxygen/examples/dds/FooDDSTopics.cxx

8.12 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>`

Inherits eprosima::rpc::transport::dds::DDSAsyncTask.Collaboration diagram for FooDDS::Foo_FooProcedureTask:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::Endpoint
eprosima::rpc::transport::AsyncTask
eprosima::rpc::transport::dds::Transport
m_asyncThread
eprosima::rpc::transport::dds::ProxyTransport
m_transport
eprosima::rpc::transport::dds::ProxyProcedureEndpoint
m_pe
eprosima::rpc::transport::dds::DDSAsyncTask
eprosima::rpc::transport::dds::AsyncThread
FooDDS::Foo_FooProcedureReply
m_FooProcedure
FooDDS::FooReply_union
m_unio
m__header
FooDDS::FooReply
m_reply
m_obj
FooDDS::Foo_FooProcedureTask
eprosima::rpc::protocol::dds::Identification
m_clientId
eprosima::rpc::protocol::dds::ReplyHeader
FooDDS::Foo_FooProcedureCallbackHandler

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.

8.12.1 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.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 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.

8.12.3 Member Function Documentation

8.12.3.1 Foo_FooProcedureCallbackHandler & FooDDS::Foo_-FooProcedureTask::getObject ()

This function returns the object used by the task.

Returns:

The object that implements the callbacks.

8.12.3.2 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.

8.12.3.3 virtual 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:

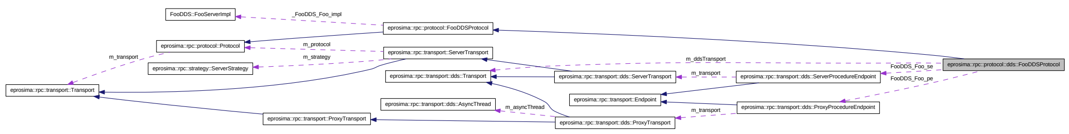
- utils/doxygen/examples/dds/FooDDSDDSAsyncSupport.h
- utils/doxygen/examples/dds/FooDDSDDSAsyncSupport.cxx

8.13 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>
```

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


FooDDS::FooServerImpl
_FooDDS_Foo_impl
eprosima::rpc::protocol::FooDDSProtocol
eprosima::rpc::protocol::Protocol
m_protocol
eprosima::rpc::transport::ServerTransport
m_transport
m_strategy
eprosima::rpc::strategy::ServerStrategy
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::Transport
eprosima::rpc::transport::dds::AsyncThread
m_asyncThread
eprosima::rpc::transport::ProxyTransport
m_ddsTransport
eprosima::rpc::transport::dds::ServerTransport
m_transport
eprosima::rpc::transport::dds::ServerProcedureEndpoint
FooDDS_Foo_se
FooDDS_Foo_pe
eprosima::rpc::protocol::dds::FooDDSProtocol
eprosima::rpc::transport::Endpoint
eprosima::rpc::transport::dds::ProxyProcedureEndpoint
m_transport
eprosima::rpc::transport::dds::ProxyTransport

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.

8.13.1 Detailed Description

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

8.13.2 Member Function Documentation

8.13.2.1 bool eprosima::rpc::protocol::dds::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.

8.13.2.2 static void eprosima::rpc::protocol::dds::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.

8.13.2.3 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 file:

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

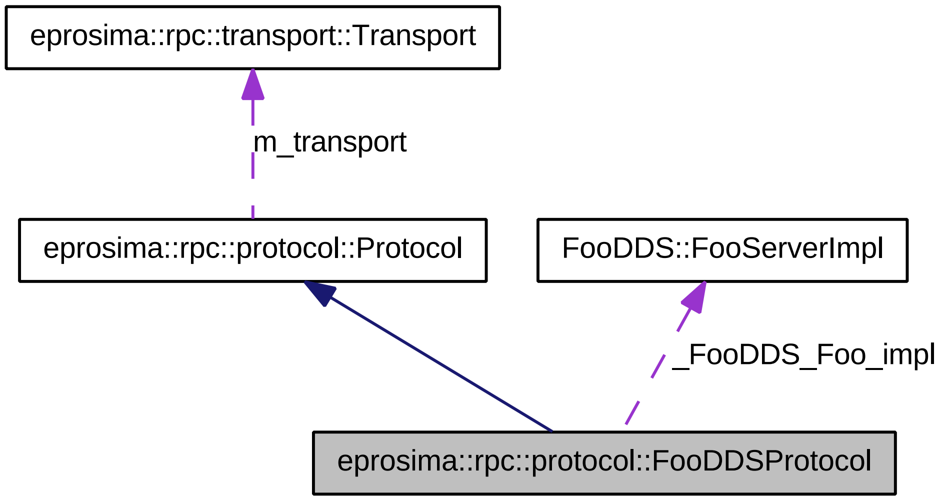
8.14 eprosima::rpc::protocol::FooDDSProtocol Class Reference

Protocol base class for the specific application.

```
#include <FooDDSProtocol.h>
```

Inherits eprosima::rpc::protocol::Protocol.

Inherited by eprosima::rpc::protocol::dds::FooDDSProtocol.Collaboration diagram for eprosima::rpc::protocol::FooDDSProtocol:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
FooDDS::FooServerImpl
_FooDDS_Foo_impl
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**

8.14.1 Detailed Description

Protocol base class for the specific application.

8.14.2 Member Function Documentation

8.14.2.1 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.

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

This method links a specific servant with the protocol.

Parameters:

impl Servant implementation.

8.14.2.3 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 file:

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

8.15 FooDDS::FooPlugin Class Reference

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

`#include <FooDDSTopicsPlugin.h>`

Classes

- class FooProcedureReplyPlugin

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

- class FooProcedureRequestPlugin

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

8.15.1 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

8.16 FooDDS::FooPlugin::FooProcedureReplyPlugin 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** ()

8.16.1 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

8.17 FooDDS::FooPlugin::FooProcedureRequestPlugin 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** ()

8.17.1 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:

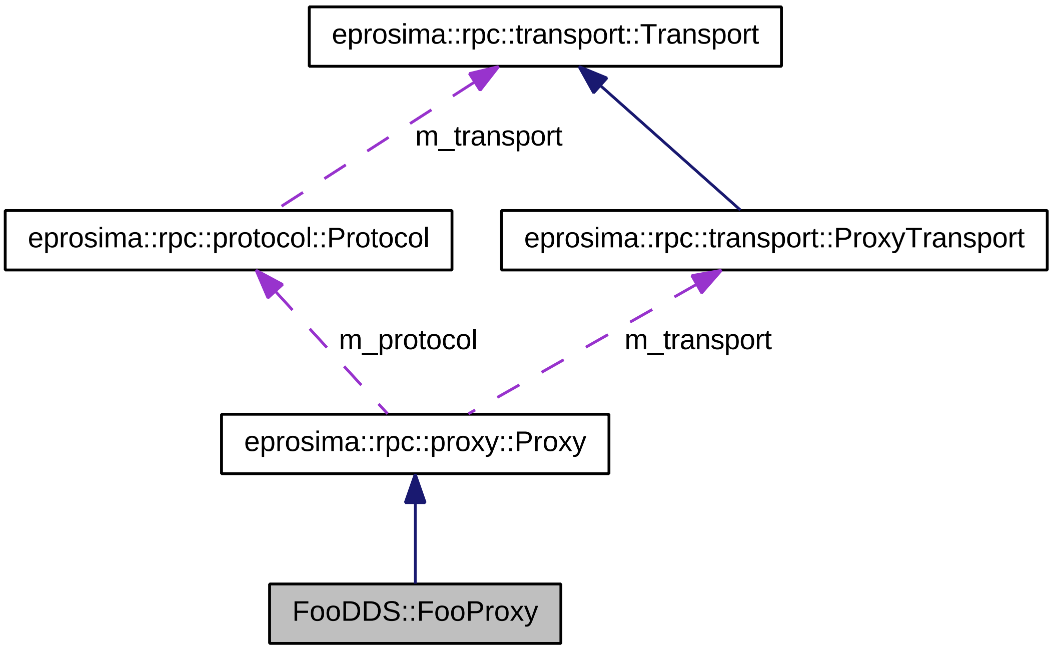
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

8.18 FooDDS::FooProxy Class Reference

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

`#include <FooDDSProxy.h>`

Inherits eprosima::rpc::proxy::Proxy.Collaboration diagram for FooDDS::FooProxy:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
eprosima::rpc::transport::ProxyTransport
m_protocol
m_transport
eprosima::rpc::proxy::Proxy
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 (Foo_FooProcedureCallbackHandler &obj)

  Proxy asynchronous method for the operation FooProcedure.

8.18.1 Detailed Description

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

8.18.2 Constructor & Destructor Documentation

8.18.2.1 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.

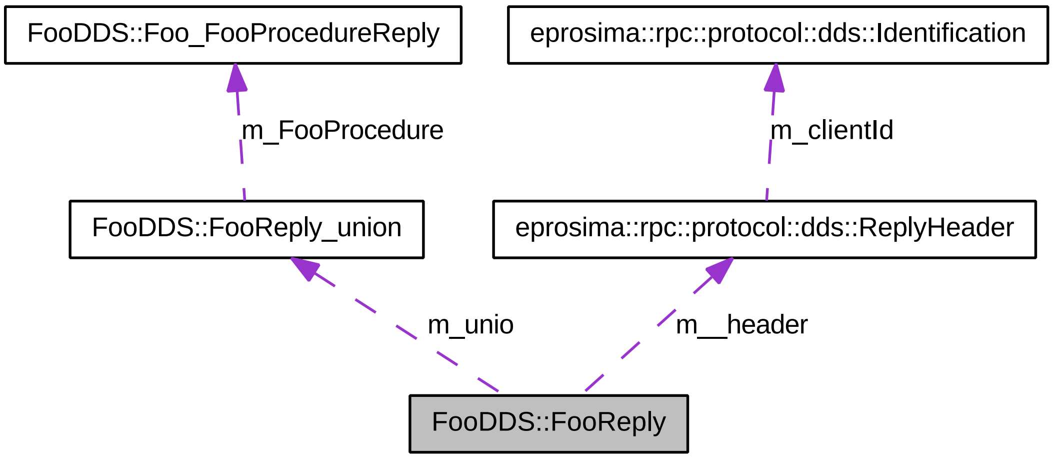
The documentation for this class was generated from the following files:

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

8.19 FooDDS::FooReply Class Reference

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

`#include <FooDDSTopics.h>`Collaboration diagram for FooDDS::FooReply:


FooDDS::Foo_FooProcedureReply
eprosima::rpc::protocol::dds::Identification
m_FooProcedure
m_clientId
FooDDS::FooReply_union
eprosima::rpc::protocol::dds::ReplyHeader
m_unio
m__header
FooDDS::FooReply

Public Member Functions

- FooReply ()

  Default constructor.

- ~FooReply ()

  Destructor.

- **FooReply** (const FooReply &x)
- FooReply (FooReply &&x)

  Move constructor.

- FooReply & operator= (const FooReply &x)

  Copy assignment.

- FooReply & operator= (FooReply &&x)

  Copy assignment.

- void _header (const eprosima::rpc::protocol::dds::ReplyHeader &__header)

  This method sets the reply header information.

- void _header (eprosima::rpc::protocol::dds::ReplyHeader &&__header)

  This method sets the reply header information.

- const eprosima::rpc::protocol::dds::ReplyHeader & _header () const

  This method returns the reply header information.

- eprosima::rpc::protocol::dds::ReplyHeader & _header ()

  This method returns the reply header information.

- void unio (const FooReply_union &_unio)

  This method sets the union that encapsulates the interface operations.

- void unio (FooReply_union &&_unio)

  This method sets the union that encapsulates the interface operations.

- const FooReply_union & unio () const

  This method sets the union that encapsulates the interface operations.

- FooReply_union & unio ()

  This method sets the union that encapsulates the interface operations.

- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function serializes an object using CDR serialization.

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.19.1 Detailed Description

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

8.19.2 Constructor & Destructor Documentation

8.19.2.1 FooDDS::FooReply::FooReply (FooReply && *x*)

Move constructor.

Parameters:

x Reference to the object FooReply that will be copied.

8.19.3 Member Function Documentation

8.19.3.1 eprosima::rpc::protocol::dds::ReplyHeader& FooDDS::FooReply::_header () `[inline]`

This method returns the reply header information.

Returns:

Reply header.

8.19.3.2 const eprosima::rpc::protocol::dds::ReplyHeader& FooDDS::FooReply::_header () const `[inline]`

This method returns the reply header information.

Returns:

Reply header.

8.19.3.3 void FooDDS::FooReply::_header (eprosima::rpc::protocol::dds::ReplyHeader && *__header*) `[inline]`

This method sets the reply header information.

Parameters:

__header Reply header.

8.19.3.4 void FooDDS::FooReply::_header (const eprosima::rpc::protocol::dds::ReplyHeader & *__header*) `[inline]`

This method sets the reply header information.

Parameters:

__header Reply header.

8.19.3.5 void FooDDS::FooReply::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.19.3.6 size_t FooDDS::FooReply::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.

8.19.3.7 size_t FooDDS::FooReply::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.

8.19.3.8 FooDDS::FooReply & FooDDS::FooReply::operator= (FooReply && *x*)

Copy assignment.

Parameters:

x Reference to the object Foo that will be copied.

8.19.3.9 FooDDS::FooReply & FooDDS::FooReply::operator= (const FooReply & *x*)

Copy assignment.

Parameters:

x Reference to the object Foo that will be copied.

8.19.3.10 void FooDDS::FooReply::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.19.3.11 FooReply_union& FooDDS::FooReply::unio () `[inline]`

This method sets the union that encapsulates the interface operations.

Returns:

Union.

8.19.3.12 const FooReply_union& FooDDS::FooReply::unio () const `[inline]`

This method sets the union that encapsulates the interface operations.

Returns:

Union.

8.19.3.13 void FooDDS::FooReply::unio (FooReply_union && *_unio*) `[inline]`

This method sets the union that encapsulates the interface operations.

Parameters:

_unio Union.

8.19.3.14 void FooDDS::FooReply::unio (const FooReply_union & *_unio*) `[inline]`

This method sets the union that encapsulates the interface operations.

Parameters:

_unio Union.

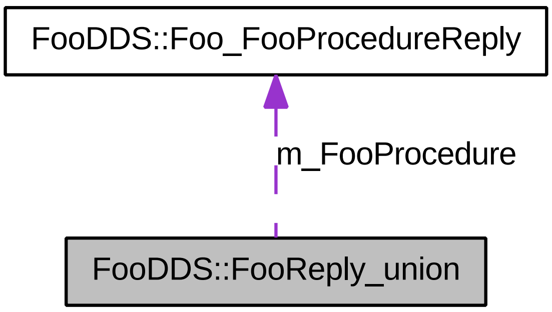
The documentation for this class was generated from the following files:

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

8.20 FooDDS::FooReply_union Class Reference

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

`#include <FooDDSTopics.h>`Collaboration diagram for FooDDS::FooReply_union:



Public Member Functions

- FooReply_union ()

  Default constructor.
- ~FooReply_union ()

  Destructor.
- FooReply_union (const FooReply_union &x)

  Copy constructor.
- FooReply_union (FooReply_union &&x)

  Move constructor.
- FooReply_union & operator= (const FooReply_union &x)

  Copy assignment.
- FooReply_union & operator= (FooReply_union &&x)

  Move assignment.
- void _d (int32_t __d)

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

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

This function returns a reference to the discriminator.

- void FooProcedure (const Foo_FooProcedureReply &_FooProcedure)

  This function copies the value in member FooProcedure.

- void FooProcedure (Foo_FooProcedureReply &&_FooProcedure)

  This function moves the value in member FooProcedure.

- const Foo_FooProcedureReply & FooProcedure () const

  This function returns a constant reference to member FooProcedure.

- Foo_FooProcedureReply & FooProcedure ()

  This function returns a reference to member FooProcedure.

- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function serializes an object using CDR serialization.

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.20.1 Detailed Description

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

8.20.2 Constructor & Destructor Documentation

8.20.2.1 FooDDS::FooReply_union::FooReply_union (const FooReply_union & *x*)

Copy constructor.

Parameters:

x Reference to the object FooReply_union that will be copied.

8.20.2.2 FooDDS::FooReply_union::FooReply_union (FooReply_union && *x*)

Move constructor.

Parameters:

x Reference to the object FooReply_union that will be copied.

8.20.3 Member Function Documentation

8.20.3.1 int32_t & FooDDS::FooReply_union::_d ()

This function returns a reference to the discriminator.

Returns:

Reference to the discriminator.

8.20.3.2 int32_t FooDDS::FooReply_union::_d () const

This function returns the value of the discriminator.

Returns:

Value of the discriminator

8.20.3.3 void FooDDS::FooReply_union::_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.

8.20.3.4 void FooDDS::FooReply_union::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.20.3.5 FooDDS::Foo_FooProcedureReply & FooDDS::FooReply_-union::FooProcedure ()

This function returns a reference to member FooProcedure.

Returns:

Reference to member FooProcedure

8.20.3.6 const FooDDS::Foo_FooProcedureReply & FooDDS::FooReply_union::FooProcedure () const

This function returns a constant reference to member FooProcedure.

Returns:

Constant reference to member FooProcedure

8.20.3.7 void FooDDS::FooReply_union::FooProcedure (FooDDS::Foo_FooProcedureReply && *_FooProcedure*)

This function moves the value in member FooProcedure.

Parameters:

_FooProcedure New value to be moved in member FooProcedure

8.20.3.8 void FooDDS::FooReply_union::FooProcedure (const Foo_FooProcedureReply & *_FooProcedure*)

This function copies the value in member FooProcedure.

Parameters:

_FooProcedure New value to be copied in member FooProcedure

8.20.3.9 size_t FooDDS::FooReply_union::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.

8.20.3.10 size_t FooDDS::FooReply_union::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.

8.20.3.11 FooDDS::FooReply_union & FooDDS::FooReply_union::operator= (FooReply_union && *x*)

Move assignment.

Parameters:

x Reference to the object FooReply_union that will be copied.

8.20.3.12 FooDDS::FooReply_union & FooDDS::FooReply_union::operator= (const FooReply_union & *x*)

Copy assignment.

Parameters:

x Reference to the object FooReply_union that will be copied.

8.20.3.13 void FooDDS::FooReply_union::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

8.21 FooDDS::FooReply_unionPlugin 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** ()

8.21.1 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

8.22 FooDDS::FooReplyDataReader Class Reference

Reply DataReader.

`#include <FooDDSTopicsPlugin.h>`

Public Member Functions

- **FooReplyDataReader** (DDSDataReader ∗impl)

8.22.1 Detailed Description

Reply DataReader.

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

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

8.23 FooDDS::FooReplyDataWriter Class Reference

Reply DataWriter.

`#include <FooDDSTopicsPlugin.h>`

Public Member Functions

- **FooReplyDataWriter** (DDSDataWriter ∗impl)

8.23.1 Detailed Description

Reply DataWriter.

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

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

8.24 FooDDS::FooReplyPlugin Class Reference

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

`#include <FooDDSTopicsPlugin.h>`

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::FooReply ∗ **create_data** (void)
- static void **destroy_data** (FooDDS::FooReply ∗sample)
- static void **copy_data** (FooDDS::FooReply ∗dst, const FooDDS::FooReply ∗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::FooReply ∗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::FooReply ∗dst, const FooDDS::FooReply ∗src)
- static RTIBool **serialize** (PRESTypePluginEndpointData endpoint_data, const FooDDS::FooReply ∗sample, struct RTICdrStream ∗stream, RTIBool serialize_encapsulation, RTIEncapsulationId encapsulation_id, RTIBool serialize_sample, void ∗endpoint_plugin_qos)

- static RTIBool **deserialize** (PRESTypePluginEndpointData endpoint_data, FooDDS::FooReply ∗∗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)

8.24.1 Detailed Description

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

8.24.2 Member Function Documentation

8.24.2.1 bool FooDDS::FooReplyPlugin::register_type (DDSDomainParticipant ∗ *participant*, const char ∗ *type_name*) `[static]`

TODO Mover al transporte

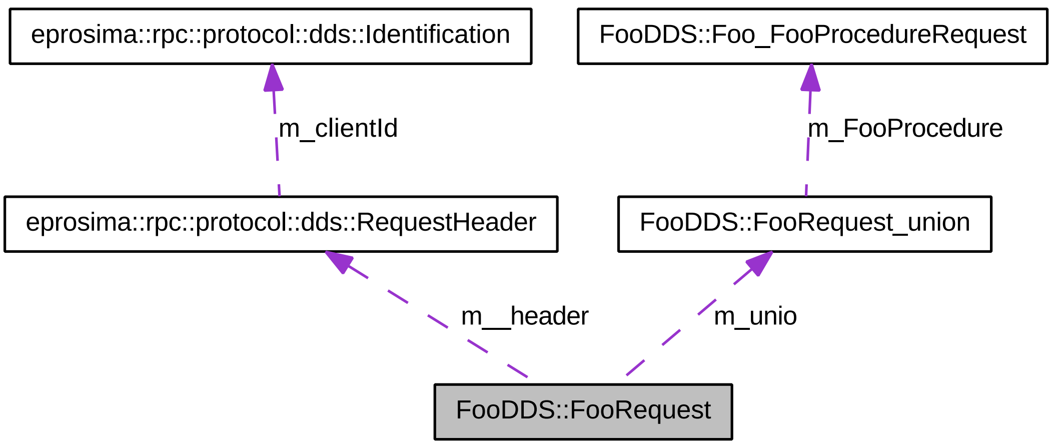
The documentation for this class was generated from the following files:

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

8.25 FooDDS::FooRequest Class Reference

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

`#include <FooDDSTopics.h>`Collaboration diagram for FooDDS::FooRequest:


eprosima::rpc::protocol::dds::Identification
FooDDS::Foo_FooProcedureRequest
m_clientId
m_FooProcedure
eprosima::rpc::protocol::dds::RequestHeader
FooDDS::FooRequest_union
m__header
m_unio
FooDDS::FooRequest

Public Member Functions

- FooRequest ()

  Default constructor.

- ~FooRequest ()

  Destructor.

- FooRequest (const FooRequest &x)

  Copy constructor.

- FooRequest (FooRequest &&x)

  Move constructor.

- FooRequest & operator= (const FooRequest &x)

  Copy assignment.

- FooRequest & operator= (FooRequest &&x)

  Copy assignment.

- void _header (const eprosima::rpc::protocol::dds::RequestHeader &__header)

  This method sets the request header information.

- void _header (eprosima::rpc::protocol::dds::RequestHeader &&__header)

  This method sets the request header information.

- const eprosima::rpc::protocol::dds::RequestHeader & _header () const

  This method returns the request header information.

- eprosima::rpc::protocol::dds::RequestHeader & _header ()

  This method returns the request header information.

- void unio (const FooRequest_union &_unio)

  This method sets the union that encapsulates the interface operations.

- void unio (FooRequest_union &&_unio)

  This method sets the union that encapsulates the interface operations.

- const FooRequest_union & unio () const

  This method returns the union that encapsulates the interface operations.

- FooRequest_union & unio ()

  This method returns the union that encapsulates the interface operations.

- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function serializes an object using CDR serialization.

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.25.1 Detailed Description

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

8.25.2 Constructor & Destructor Documentation

8.25.2.1 FooDDS::FooRequest::FooRequest (const FooRequest & *x*)

Copy constructor.

Parameters:

x Reference to the object FooRequest that will be copied.

8.25.2.2 FooDDS::FooRequest::FooRequest (FooRequest && *x*)

Move constructor.

Parameters:

x Reference to the object FooRequest that will be copied.

8.25.3 Member Function Documentation

8.25.3.1 eprosima::rpc::protocol::dds::RequestHeader& FooDDS::FooRequest::_header () `[inline]`

This method returns the request header information.

Returns:

Request header.

8.25.3.2 const eprosima::rpc::protocol::dds::RequestHeader& FooDDS::FooRequest::_header () const `[inline]`

This method returns the request header information.

Returns:

Request header.

8.25.3.3 void FooDDS::FooRequest::_header (eprosima::rpc::protocol::dds::RequestHeader && *__header*) `[inline]`

This method sets the request header information.

Parameters:

__header Request header.

8.25.3.4 void FooDDS::FooRequest::_header (const eprosima::rpc::protocol::dds::RequestHeader & *__header*) `[inline]`

This method sets the request header information.

Parameters:

__header Request header.

8.25.3.5 void FooDDS::FooRequest::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.25.3.6 size_t FooDDS::FooRequest::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.

8.25.3.7 size_t FooDDS::FooRequest::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.

8.25.3.8 FooDDS::FooRequest & FooDDS::FooRequest::operator= (FooRequest && *x*)

Copy assignment.

Parameters:

x Reference to the object Foo that will be copied.

8.25.3.9 FooDDS::FooRequest & FooDDS::FooRequest::operator= (const FooRequest & *x*)

Copy assignment.

Parameters:

x Reference to the object Foo that will be copied.

8.25.3.10 void FooDDS::FooRequest::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.25.3.11 FooRequest_union& FooDDS::FooRequest::unio () `[inline]`

This method returns the union that encapsulates the interface operations.

Returns:

Union.

8.25.3.12 const FooRequest_union& FooDDS::FooRequest::unio () const `[inline]`

This method returns the union that encapsulates the interface operations.

Returns:

Union.

8.25.3.13 void FooDDS::FooRequest::unio (FooRequest_union && *_unio*) `[inline]`

This method sets the union that encapsulates the interface operations.

Parameters:

_unio Union.

8.25.3.14 void FooDDS::FooRequest::unio (const FooRequest_union & *_unio*) `[inline]`

This method sets the union that encapsulates the interface operations.

Parameters:

_unio Union.

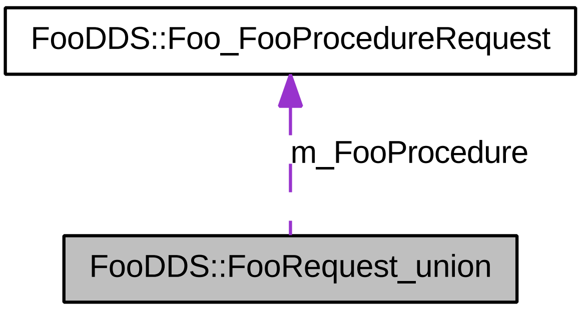
The documentation for this class was generated from the following files:

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

8.26 FooDDS::FooRequest_union Class Reference

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

`#include <FooDDSTopics.h>`Collaboration diagram for FooDDS::FooRequest_union:



Public Member Functions

- FooRequest_union ()

  Default constructor.
- ~FooRequest_union ()

  Destructor.
- FooRequest_union (const FooRequest_union &x)

  Copy constructor.
- FooRequest_union (FooRequest_union &&x)

  Move constructor.
- FooRequest_union & operator= (const FooRequest_union &x)

  Copy assignment.
- FooRequest_union & operator= (FooRequest_union &&x)

  Move assignment.
- void _d (int32_t __d)

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

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

This function returns a reference to the discriminator.

- void FooProcedure (const Foo_FooProcedureRequest &_FooProcedure)

  This function copies the value in member FooProcedure.

- void FooProcedure (Foo_FooProcedureRequest &&_FooProcedure)

  This function moves the value in member FooProcedure.

- const Foo_FooProcedureRequest & FooProcedure () const

  This function returns a constant reference to member FooProcedure.

- Foo_FooProcedureRequest & FooProcedure ()

  This function returns a reference to member FooProcedure.

- size_t getSerializedSize (size_t current_alignment=0) const

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

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

  This function serializes an object using CDR serialization.

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

  This function deserializes an object using CDR serialization.

Static Public Member Functions

- static size_t getMaxCdrSerializedSize (size_t current_alignment=0)

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

8.26.1 Detailed Description

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

8.26.2 Constructor & Destructor Documentation

8.26.2.1 FooDDS::FooRequest_union::FooRequest_union (const FooRequest_union & *x*)

Copy constructor.

Parameters:

x Reference to the object FooRequest_union that will be copied.

8.26.2.2 FooDDS::FooRequest_union::FooRequest_union (FooRequest_union && *x*)

Move constructor.

Parameters:

x Reference to the object FooRequest_union that will be copied.

8.26.3 Member Function Documentation

8.26.3.1 int32_t & FooDDS::FooRequest_union::_d ()

This function returns a reference to the discriminator.

Returns:

Reference to the discriminator.

8.26.3.2 int32_t FooDDS::FooRequest_union::_d () const

This function returns the value of the discriminator.

Returns:

Value of the discriminator

8.26.3.3 void FooDDS::FooRequest_union::_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.

8.26.3.4 void FooDDS::FooRequest_union::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes an object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.26.3.5 FooDDS::Foo_FooProcedureRequest & FooDDS::FooRequest_-union::FooProcedure ()

This function returns a reference to member FooProcedure.

Returns:

Reference to member FooProcedure

8.26.3.6 const FooDDS::Foo_FooProcedureRequest & FooDDS::FooRequest_union::FooProcedure () const

This function returns a constant reference to member FooProcedure.

Returns:

Constant reference to member FooProcedure

8.26.3.7 void FooDDS::FooRequest_union::FooProcedure (FooDDS::Foo_FooProcedureRequest && *_FooProcedure*)

This function moves the value in member FooProcedure.

Parameters:

_FooProcedure New value to be moved in member FooProcedure

8.26.3.8 void FooDDS::FooRequest_union::FooProcedure (const Foo_FooProcedureRequest & *_FooProcedure*)

This function copies the value in member FooProcedure.

Parameters:

_FooProcedure New value to be copied in member FooProcedure

8.26.3.9 size_t FooDDS::FooRequest_union::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.

8.26.3.10 size_t FooDDS::FooRequest_union::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.

8.26.3.11 FooDDS::FooRequest_union & FooDDS::FooRequest_-union::operator= (FooRequest_union && *x*)

Move assignment.

Parameters:

x Reference to the object FooRequest_union that will be copied.

8.26.3.12 FooDDS::FooRequest_union & FooDDS::FooRequest_-union::operator= (const FooRequest_union & *x*)

Copy assignment.

Parameters:

x Reference to the object FooRequest_union that will be copied.

8.26.3.13 void FooDDS::FooRequest_union::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

8.27 FooDDS::FooRequest_unionPlugin 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** ()

8.27.1 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

8.28 FooDDS::FooRequestDataReader Class Reference

Request DataReader.

`#include <FooDDSTopicsPlugin.h>`

Public Member Functions

- **FooRequestDataReader** (DDSDataReader ∗impl)

8.28.1 Detailed Description

Request DataReader.

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

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

8.29 FooDDS::FooRequestDataWriter Class Reference

Request DataWriter.

```
#include <FooDDSTopicsPlugin.h>
```

Public Member Functions

- **FooRequestDataWriter** (DDSDataWriter ∗impl)

8.29.1 Detailed Description

Request DataWriter.

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

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

8.30 FooDDS::FooRequestPlugin Class Reference

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

`#include <FooDDSTopicsPlugin.h>`

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::FooRequest ∗ **create_data** (void)
- static void **destroy_data** (FooDDS::FooRequest ∗sample)
- static void **copy_data** (FooDDS::FooRequest ∗dst, const FooDDS::FooRequest ∗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::FooRequest ∗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::FooRequest ∗dst, const FooDDS::FooRequest ∗src)
- static RTIBool **serialize** (PRESTypePluginEndpointData endpoint_data, const FooDDS::FooRequest ∗sample, struct RTICdrStream ∗stream, RTIBool serialize_encapsulation, RTIEncapsulationId encapsulation_id, RTIBool serialize_sample, void ∗endpoint_plugin_qos)

- static RTIBool **deserialize** (PRESTypePluginEndpointData endpoint_data, FooDDS::FooRequest ∗∗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)

8.30.1 Detailed Description

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

8.30.2 Member Function Documentation

8.30.2.1 bool FooDDS::FooRequestPlugin::register_type (DDSDomainParticipant ∗ *participant*, const char ∗ *type_name*) `[static]`

TODO Mover al transporte

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

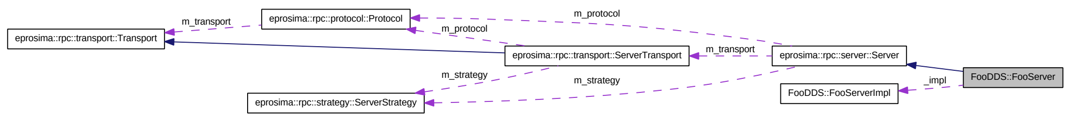
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.h
- utils/doxygen/examples/dds/FooDDSTopicsPlugin.cxx

8.31 FooDDS::FooServer Class Reference

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

`#include <FooDDSServer.h>`

Inherits eprosima::rpc::server::Server.Collaboration diagram for FooDDS::FooServer:


m_transport
eprosima::rpc::protocol::Protocol
m_protocol
eprosima::rpc::transport::Transport
m_protocol
eprosima::rpc::transport::ServerTransport
m_transport
eprosima::rpc::server::Server
FooDDS::FooServer
m_strategy
m_strategy
_impl
FooDDS::FooServerImpl
eprosima::rpc::strategy::ServerStrategy

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.

8.31.1 Detailed Description

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

8.31.2 Constructor & Destructor Documentation

8.31.2.1 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

8.32 FooDDS::FooServerImpl Class Reference

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

```
#include <FooDDSServerImpl.h>
```

Public Member Functions

- FooServerImpl ()

  The default constructor.

- virtual ~FooServerImpl ()

  Destructor.

- virtual void FooProcedure ()=0

  Skeleton of the operation FooProcedure.

8.32.1 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

8.33 eprosima::rpc::protocol::dds::Identification Class Reference

This class is used to identify clients.

`#include <MessageHeader.h>`

Public Member Functions

- Identification ()

  Default constructor.
- Identification (const Identification &id)

  Copy constructor.
- Identification (Identification &&id)

  Copy constructor.
- ~Identification ()

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

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

  Copy assignment.
- void value_1 (uint32_t _value_1)

  This function sets the first value of the client identifier.
- uint32_t value_1 () const

  This function returns the first value of the client identifier.
- uint32_t & value_1 ()

  This function returns the first value of the client identifier.
- void value_2 (uint32_t _value_2)

  This function sets the second value of the client identifier.
- uint32_t value_2 () const

  This function returns the second value of the client identifier.
- uint32_t & value_2 ()

  This function returns the second value of the client identifier.
- void value_3 (uint32_t _value_3)

This function sets the third value of the client identifier.

- uint32_t value_3 () const

  This function returns the third value of the client identifier.

- uint32_t & value_3 ()

  This function returns the third value of the client identifier.

- void value_4 (uint32_t _value_4)

  This function sets the fourth value of the client identifier.

- uint32_t value_4 () const

  This function returns the fourth value of the client identifier.

- uint32_t & value_4 ()

  This function returns the fourth value of the client identifier.

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

  This function serializes the Identification object using CDR serialization.

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

  This function deserializes the Identification object using CDR serialization.

Static Public Member Functions

- static unsigned int getMaxCdrSerializedSize (unsigned int current_alignment)

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

8.33.1 Detailed Description

This class is used to identify clients.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 eprosima::rpc::protocol::dds::Identification::Identification (const Identification & *id*)

Copy constructor.

Parameters:

id Identification object to be copied.

8.33.2.2 eprosima::rpc::protocol::dds::Identification::Identification (Identification && *id*)

Copy constructor.

Parameters:

id Identification object to be copied.

8.33.3 Member Function Documentation

8.33.3.1 void eprosima::rpc::protocol::dds::Identification::deserialize (eprosima::fastcdr::Cdr & *cdr*)

This function deserializes the Identification object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.33.3.2 static unsigned int eprosima::rpc::protocol::dds::Identification::getMaxCdrSerializedSize (unsigned int *current_alignment*) `[static]`

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

Parameters:

current_alignment Buffer alignment.

Returns:

Maximum serialized size.

8.33.3.3 Identification& eprosima::rpc::protocol::dds::Identification::operator= (Identification && *id*)

Copy assignment.

Parameters:

id Identification object to be copied.

8.33.3.4 Identification& eprosima::rpc::protocol::dds::Identification::operator= (const Identification & *id*)

Copy assignment.

Parameters:

id Identification object to be copied.

8.33.3.5 void eprosima::rpc::protocol::dds::Identification::serialize (eprosima::fastcdr::Cdr & *cdr*) const

This function serializes the Identification object using CDR serialization.

Parameters:

cdr CDR serialization object.

8.33.3.6 uint32_t& eprosima::rpc::protocol::dds::Identification::value_1 () `[inline]`

This function returns the first value of the client identifier.

Returns:

First value of the client identifier.

8.33.3.7 uint32_t eprosima::rpc::protocol::dds::Identification::value_1 () const `[inline]`

This function returns the first value of the client identifier.

Returns:

First value of the client identifier.

8.33.3.8 void eprosima::rpc::protocol::dds::Identification::value_1 (uint32_t *_value_1*) `[inline]`

This function sets the first value of the client identifier.

Parameters:

_value_1 First value of the client identifier.

8.33.3.9 uint32_t& eprosima::rpc::protocol::dds::Identification::value_2 () `[inline]`

This function returns the second value of the client identifier.

Returns:

Second value of the client identifier.

8.33.3.10 uint32_t eprosima::rpc::protocol::dds::Identification::value_2 () const `[inline]`

This function returns the second value of the client identifier.

Returns:

Second value of the client identifier.

8.33.3.11 void eprosima::rpc::protocol::dds::Identification::value_2 (uint32_t *_value_2*) `[inline]`

This function sets the second value of the client identifier.

Parameters:

_value_2 Second value of the client identifier.

8.33.3.12 uint32_t& eprosima::rpc::protocol::dds::Identification::value_3 () `[inline]`

This function returns the third value of the client identifier.

Returns:

Third value of the client identifier.

8.33.3.13 uint32_t eprosima::rpc::protocol::dds::Identification::value_3 () const `[inline]`

This function returns the third value of the client identifier.

Returns:

Third value of the client identifier.

8.33.3.14 void eprosima::rpc::protocol::dds::Identification::value_3 (uint32_t *_value_3*) `[inline]`

This function sets the third value of the client identifier.

Parameters:

_value_3 Third value of the client identifier.

8.33.3.15 uint32_t& eprosima::rpc::protocol::dds::Identification::value_4 () `[inline]`

This function returns the fourth value of the client identifier.

Returns:

Fourth value of the client identifier.

8.33.3.16 uint32_t eprosima::rpc::protocol::dds::Identification::value_4 () const `[inline]`

This function returns the fourth value of the client identifier.

Returns:

Fourth value of the client identifier.

8.33.3.17 void eprosima::rpc::protocol::dds::Identification::value_4 (uint32_t *_value_4*) `[inline]`

This function sets the fourth value of the client identifier.

Parameters:

_value_4 Fourth value of the client identifier.

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

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

8.34 eprosima::rpc::protocol::dds::IdentificationPlugin Class Reference

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

```
#include <MessageHeaderPlugin.h>
```

Static Public Member Functions

- static DDS_TypeCode ∗ get_typecode ()

  This function returns the TypeCode.

8.34.1 Detailed Description

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

8.34.2 Member Function Documentation

8.34.2.1 static DDS_TypeCode∗ eprosima::rpc::protocol::dds::IdentificationPlugin::get_typecode () [static]

This function returns the TypeCode.

Returns:

The TypeCode.

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

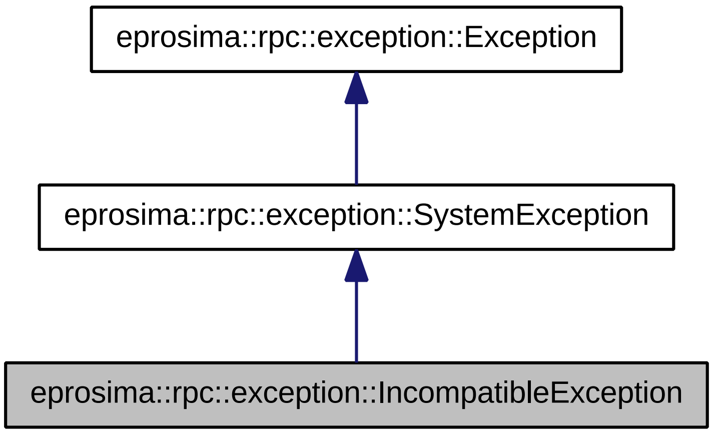
- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

8.35 eprosima::rpc::exception::IncompatibleException Class Reference

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

`#include <IncompatibleException.h>`

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::IncompatibleException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::IncompatibleException

Public Member Functions

- IncompatibleException (const std::string &message)

  Default constructor.

- IncompatibleException (std::string &&message)

  Default constructor.

- IncompatibleException (const IncompatibleException &ex)

  Default copy constructor.

- IncompatibleException (IncompatibleException &&ex)

  Default move constructor.

- IncompatibleException & operator= (const IncompatibleException &ex)

  Assigment operation.

- IncompatibleException & operator= (IncompatibleException &&ex)

  Assigment operation.

- virtual ∼IncompatibleException () throw ()

Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.35.1 Detailed Description

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

8.35.2 Constructor & Destructor Documentation

8.35.2.1 eprosima::rpc::exception::IncompatibleException::IncompatibleException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.35.2.2 eprosima::rpc::exception::IncompatibleException::IncompatibleException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.35.2.3 eprosima::rpc::exception::IncompatibleException::IncompatibleException (const IncompatibleException & *ex*)

Default copy constructor.

Parameters:

ex IncompatibleException that will be copied.

8.35.2.4 eprosima::rpc::exception::IncompatibleException::IncompatibleException (IncompatibleException && *ex*)

Default move constructor.

Parameters:

ex IncompatibleException that will be moved.

8.35.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex IncompatibleException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

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

Assigment operation.

Parameters:

ex IncompatibleException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

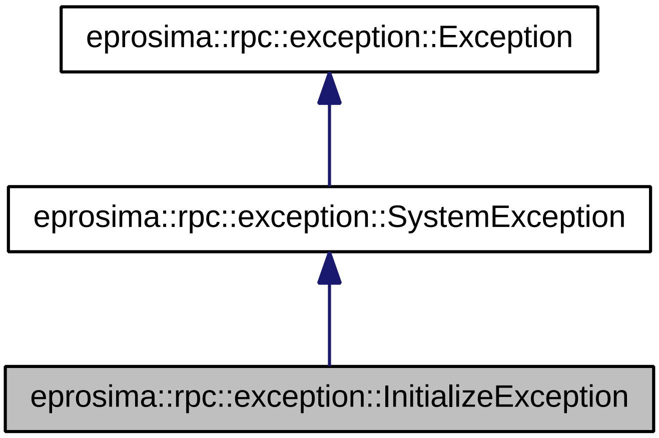
- includetmp/rpcdds/exceptions/IncompatibleException.h

8.36 eprosima::rpc::exception::InitializeException Class Reference

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

```
#include <InitializeException.h>
```

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::InitializeException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::InitializeException

Public Member Functions

- InitializeException (const std::string &message)

  Default constructor.

- InitializeException (std::string &&message)

  Default constructor.

- InitializeException (const InitializeException &ex)

  Default copy constructor.

- InitializeException (InitializeException &&ex)

  Default move constructor.

- InitializeException & operator= (const InitializeException &ex)

  Assigment operation.

- InitializeException & operator= (InitializeException &&ex)

  Assigment operation.

- virtual ∼InitializeException () throw ()

Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.36.1 Detailed Description

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

8.36.2 Constructor & Destructor Documentation

8.36.2.1 eprosima::rpc::exception::InitializeException::InitializeException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.36.2.2 eprosima::rpc::exception::InitializeException::InitializeException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.36.2.3 eprosima::rpc::exception::InitializeException::InitializeException (const InitializeException & *ex*)

Default copy constructor.

Parameters:

ex InitializeException that will be copied.

8.36.2.4 eprosima::rpc::exception::InitializeException::InitializeException (InitializeException && *ex*)

Default move constructor.

Parameters:

ex InitializeException that will be moved.

8.36.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex InitializeException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

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

Assigment operation.

Parameters:

ex InitializeException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

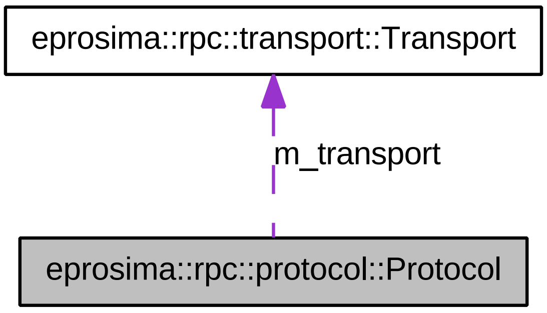
- includetmp/rpcdds/exceptions/InitializeException.h

8.37 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>`

Inherited by eprosima::rpc::protocol::FooDDSProtocol.Collaboration diagram for eprosima::rpc::protocol::Protocol:


eprosima::rpc::transport::Transport
m_transport
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.

8.37.1 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.

8.37.2 Member Function Documentation

8.37.2.1 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.

8.37.2.2 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.

8.37.2.3 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:

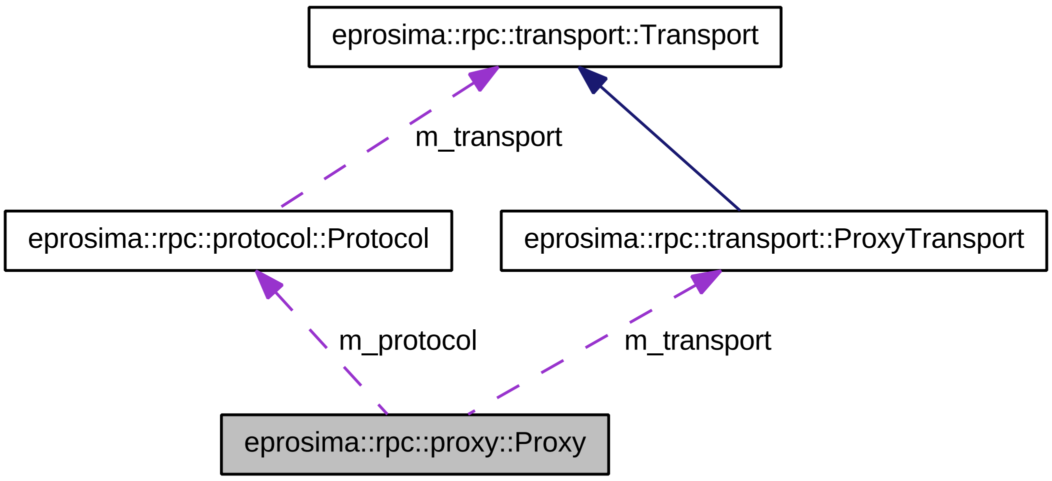
- includetmp/rpcdds/protocols/Protocol.h

8.38 eprosima::rpc::proxy::Proxy Class Reference

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

`#include <Proxy.h>`

Inherited by FooDDS::FooProxy.Collaboration diagram for eprosima::rpc::proxy::Proxy:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
eprosima::rpc::transport::ProxyTransport
m_protocol
m_transport
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.

8.38.1 Detailed Description

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

8.38.2 Constructor & Destructor Documentation

8.38.2.1 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.

8.38.3 Member Function Documentation

8.38.3.1 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

8.38.3.2 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:

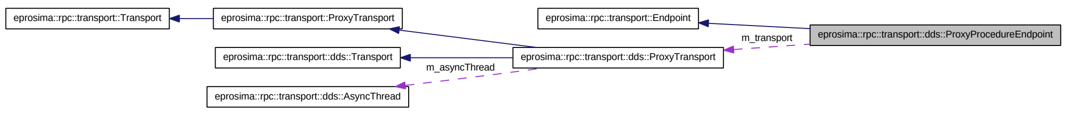
- includetmp/rpcdds/client/Proxy.h

8.39 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>`

Inherits eprosima::rpc::transport::Endpoint.Collaboration diagram for eprosima::rpc::transport::dds::ProxyProcedureEndpoint:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::Endpoint
eprosima::rpc::transport::dds::ProxyProcedureEndpoint
m_transport
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::dds::ProxyTransport
m_asyncThread
eprosima::rpc::transport::dds::AsyncThread

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 DDSAsync-Task object will be invoked when the response arrives.

- void freeQuery (DDS::QueryCondition ∗query)

  Frees a DDS query condition.

- eprosima::rpc::ReturnMessage takeReply (void ∗reply, DDS::QueryCondition ∗query)

This function takes a sample from the datareader.

8.39.1 Detailed Description

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

8.39.2 Constructor & Destructor Documentation

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

Default constructor.

Parameters:

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

8.39.3 Member Function Documentation

8.39.3.1 void eprosima::rpc::transport::dds::ProxyProcedureEndpoint::freeQuery (DDS::QueryCondition ∗ *query*)

Frees a DDS query condition.

Parameters:

query Query condition to free.

8.39.3.2 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

8.39.3.3 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.

8.39.3.4 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 DDSAsync-Task 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

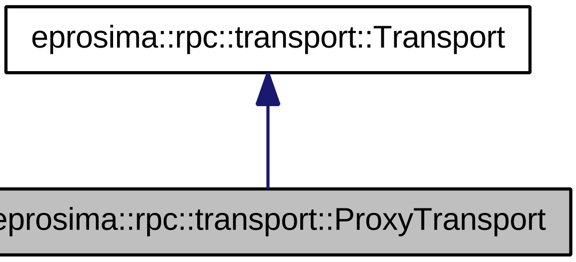
8.40 eprosima::rpc::transport::ProxyTransport Class Reference

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

`#include <ProxyTransport.h>`

Inherits eprosima::rpc::transport::Transport.

Inherited by eprosima::rpc::transport::dds::ProxyTransport.Collaboration diagram for eprosima::rpc::transport::ProxyTransport:


eprosima::rpc::transport::Transport
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 &data-ToRead)=0

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

8.40.1 Detailed Description

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

8.40.2 Member Function Documentation

8.40.2.1 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.

8.40.2.2 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.

8.40.2.3 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.

8.40.2.4 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

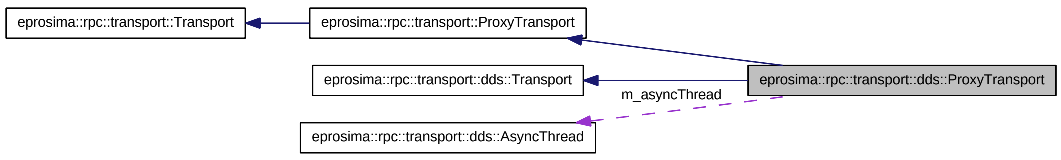
8.41 eprosima::rpc::transport::dds::ProxyTransport Class Reference

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

`#include <ProxyTransport.h>`

Inherits eprosima::rpc::transport::ProxyTransport, and eprosima::rpc::transport::dds::Transport.

Inherited by eprosima::rpc::transport::dds::TCPProxyTransport, and eprosima::rpc::transport::dds::UDPProxyTransport.Collaboration diagram for eprosima::rpc::transport::dds::ProxyTransport:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::dds::ProxyTransport
m_asyncThread
eprosima::rpc::transport::dds::AsyncThread

Public Member Functions

- virtual ∼ProxyTransport ()

  Default destructor.

- virtual const char ∗ getType () const

  This abstract function returns the type of the transport. This function has to be implemented by the child classes.

- std::string & getRemoteServiceName ()

  This function returns the DDS service name.

- long getTimeout ()

  This function gets the timeout value.

- 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.

- int addAsyncTask (DDS::QueryCondition ∗query, DDSAsyncTask ∗task, long timeout)

  This function adds a asynchronous task to the asynchronous thread.

- void deleteAssociatedAsyncTasks (ProxyProcedureEndpoint ∗pe)

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

Protected Member Functions

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)=0

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

- ProxyTransport (std::string &remoteServiceName, int domainId=0, long milliseconds=10000L)

  Default constructor.

8.41.1 Detailed Description

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

8.41.2 Constructor & Destructor Documentation

8.41.2.1 eprosima::rpc::transport::dds::ProxyTransport::ProxyTransport (std::string & *remoteServiceName*, int *domainId* = `0`, long *milliseconds* = `10000L`) `[protected]`

Default constructor.

Parameters:

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

8.41.3 Member Function Documentation

8.41.3.1 int eprosima::rpc::transport::dds::ProxyTransport::addAsyncTask (DDS::QueryCondition ∗ *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.

8.41.3.2 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.

8.41.3.3 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.

8.41.3.4 std::string& eprosima::rpc::transport::dds::ProxyTransport::getRemoteServiceName ()

This function returns the DDS service name.

Returns:

DDS service name.

8.41.3.5 long eprosima::rpc::transport::dds::ProxyTransport::getTimeout ()

This function gets the timeout value.

Returns:

Timeout value.

8.41.3.6 virtual int eprosima::rpc::transport::dds::ProxyTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *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::TCPProxyTransport, and eprosima::rpc::transport::dds::UDPProxyTransport.

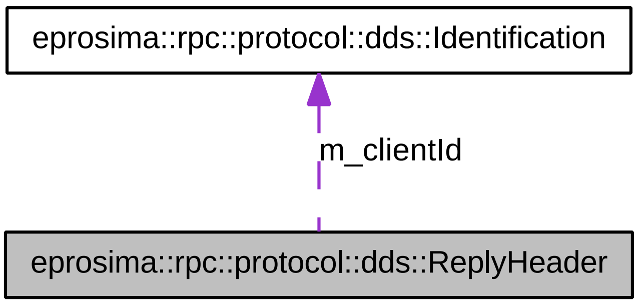
The documentation for this class was generated from the following file:

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

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

Header information used in all generated reply topics.

`#include <MessageHeader.h>`Collaboration diagram for eprosima::rpc::protocol::dds::ReplyHeader:


eprosima::rpc::protocol::dds::Identification
m_clientId
eprosima::rpc::protocol::dds::ReplyHeader

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 clientId (const Identification &_clientId)

  This function sets the client identifier.

- void clientId (Identification &&_clientId)

  This function sets the client identifier.

- const Identification & clientId () const

This function returns the client identifier.

- Identification & clientId ()

  This function returns the client identifier.

- void requestSequenceNumber (uint32_t _requestSequenceNumber)

  This function sets the request sequence number.

- uint32_t requestSequenceNumber () const

  This function returns the request sequence number.

- uint32_t & requestSequenceNumber ()

  This function returns the request sequence number.

- void retCode (int32_t _retCode)

  This function sets the server return code.

- int32_t retCode () const

  This function returns the server return code.

- int32_t & retCode ()

  This function returns the server return code.

- void retMsg (const std::string &_retMsg)

  This function sets the server return message.

- void retMsg (std::string &&_retMsg)

  This function sets the server return message.

- const std::string & retMsg () const

  This function returns the server return message.

- std::string & retMsg ()

  This function returns the server return message.

- 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 unsigned int getMaxCdrSerializedSize (unsigned int current_alignment)

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

8.42.1 Detailed Description

Header information used in all generated reply topics.

8.42.2 Constructor & Destructor Documentation

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

Copy constructor.

Parameters:

header	ReplyHeader object to be copied.

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

Copy constructor.

Parameters:

header	ReplyHeader object to be copied.

8.42.3 Member Function Documentation

8.42.3.1 Identification& eprosima::rpc::protocol::dds::ReplyHeader::clientId () `[inline]`

This function returns the client identifier.

Returns:

Client identifier

8.42.3.2 const Identification& eprosima::rpc::protocol::dds::ReplyHeader::clientId () const `[inline]`

This function returns the client identifier.

Returns:

Client identifier

8.42.3.3 void eprosima::rpc::protocol::dds::ReplyHeader::clientId (Identification && *_clientId*) `[inline]`

This function sets the client identifier.

Parameters:

_clientId Client identifier

8.42.3.4 void eprosima::rpc::protocol::dds::ReplyHeader::clientId (const Identification & *_clientId*) `[inline]`

This function sets the client identifier.

Parameters:

_clientId Client identifier

8.42.3.5 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.

8.42.3.6 static unsigned int eprosima::rpc::protocol::dds::ReplyHeader::getMaxCdrSerializedSize (unsigned int *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.

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

Copy assignment.

Parameters:

header ReplyHeader object to be copied.

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

Copy assignment.

Parameters:

header ReplyHeader object to be copied.

8.42.3.9 uint32_t& eprosima::rpc::protocol::dds::ReplyHeader::requestSequenceNumber () `[inline]`

This function returns the request sequence number.

Returns:

Request sequence number

8.42.3.10 uint32_t eprosima::rpc::protocol::dds::ReplyHeader::requestSequenceNumber () const `[inline]`

This function returns the request sequence number.

Returns:

Request sequence number

8.42.3.11 void eprosima::rpc::protocol::dds::ReplyHeader::requestSequenceNumber (uint32_t *_requestSequenceNumber*) `[inline]`

This function sets the request sequence number.

Parameters:

_requestSequenceNumber Request sequence number

8.42.3.12 int32_t& eprosima::rpc::protocol::dds::ReplyHeader::retCode () `[inline]`

This function returns the server return code.

Returns:

Server return code

8.42.3.13 int32_t eprosima::rpc::protocol::dds::ReplyHeader::retCode () const `[inline]`

This function returns the server return code.

Returns:

Server return code

8.42.3.14 void eprosima::rpc::protocol::dds::ReplyHeader::retCode (int32_t *_retCode*) `[inline]`

This function sets the server return code.

Parameters:

_retCode Server return code

8.42.3.15 std::string& eprosima::rpc::protocol::dds::ReplyHeader::retMsg () `[inline]`

This function returns the server return message.

Returns:

Server return message

8.42.3.16 const std::string& eprosima::rpc::protocol::dds::ReplyHeader::retMsg () const `[inline]`

This function returns the server return message.

Returns:

Server return message

8.42.3.17 void eprosima::rpc::protocol::dds::ReplyHeader::retMsg (std::string && *_retMsg*) `[inline]`

This function sets the server return message.

Parameters:

_retMsg Server return message

8.42.3.18 void eprosima::rpc::protocol::dds::ReplyHeader::retMsg (const std::string & *_retMsg*) `[inline]`

This function sets the server return message.

Parameters:

_retMsg Server return message

8.42.3.19 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

8.43 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.

8.43.1 Detailed Description

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

8.43.2 Member Function Documentation

8.43.2.1 static DDS_TypeCode∗ eprosima::rpc::protocol::dds::ReplyHeaderPlugin::get_typecode () `[static]`

This function returns the TypeCode.

Returns:

The TypeCode.

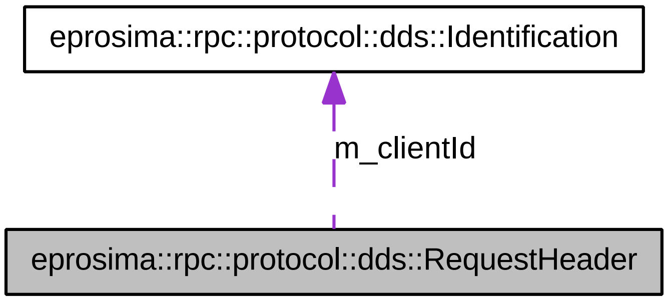
The documentation for this class was generated from the following file:

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

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

Header information used in all generated request topics.

`#include <MessageHeader.h>`Collaboration diagram for eprosima::rpc::protocol::dds::RequestHeader:


eprosima::rpc::protocol::dds::Identification
m_clientId
eprosima::rpc::protocol::dds::RequestHeader

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 clientId (const Identification &_clientId)
  This function sets the client identifier.
- void clientId (Identification &&_clientId)
  This function sets the client identifier.
- const Identification & clientId () const

This function returns the client identifier.

- Identification & clientId ()

  This function returns the client identifier.

- void remoteServiceName (const char ∗_remoteServiceName)

  This function sets the server service name.

- const char ∗ remoteServiceName () const

  This function returns the server service name.

- void requestSequenceNumber (uint32_t _requestSequenceNumber)

  This function sets the request sequence number.

- uint32_t requestSequenceNumber () const

  This function returns the request sequence number.

- uint32_t & requestSequenceNumber ()

  This function returns the request sequence number.

- 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 unsigned int getMaxCdrSerializedSize (unsigned int current_alignment)

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

8.44.1 Detailed Description

Header information used in all generated request topics.

8.44.2 Constructor & Destructor Documentation

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

Copy constructor.

Parameters:

header RequestHeader object to be copied.

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

Copy constructor.

Parameters:

header RequestHeader object to be copied.

8.44.3 Member Function Documentation

8.44.3.1 Identification& eprosima::rpc::protocol::dds::RequestHeader::clientId () `[inline]`

This function returns the client identifier.

Returns:

Client identifier

8.44.3.2 const Identification& eprosima::rpc::protocol::dds::RequestHeader::clientId () const `[inline]`

This function returns the client identifier.

Returns:

Client identifier

8.44.3.3 void eprosima::rpc::protocol::dds::RequestHeader::clientId (Identification && *_clientId*) `[inline]`

This function sets the client identifier.

Parameters:

_clientId Client identifier

8.44.3.4 void eprosima::rpc::protocol::dds::RequestHeader::clientId (const Identification & *_clientId*) `[inline]`

This function sets the client identifier.

Parameters:

_clientId Client identifier

8.44.3.5 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.

8.44.3.6 static unsigned int eprosima::rpc::protocol::dds::RequestHeader::getMaxCdrSerializedSize (unsigned int *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.

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

Copy assignment.

Parameters:

header RequestHeader object to be copied.

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

Copy assignment.

Parameters:

header RequestHeader object to be copied.

8.44.3.9 const char∗ eprosima::rpc::protocol::dds::RequestHeader::remoteServiceName () const `[inline]`

This function returns the server service name.

Returns:

Server service name.

8.44.3.10 void eprosima::rpc::protocol::dds::RequestHeader::remoteServiceName (const char ∗ *_remoteServiceName*) `[inline]`

This function sets the server service name.

Parameters:

_remoteServiceName Server service name.

8.44.3.11 uint32_t& eprosima::rpc::protocol::dds::RequestHeader::requestSequenceNumber () `[inline]`

This function returns the request sequence number.

Returns:

Request sequence number

8.44.3.12 uint32_t eprosima::rpc::protocol::dds::RequestHeader::requestSequenceNumber () const `[inline]`

This function returns the request sequence number.

Returns:

Request sequence number

8.44.3.13 void eprosima::rpc::protocol::dds::RequestHeader::requestSequenceNumber (uint32_t *_requestSequenceNumber*) `[inline]`

This function sets the request sequence number.

Parameters:

_requestSequenceNumber Request sequence number

8.44.3.14 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

8.45 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.

8.45.1 Detailed Description

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

8.45.2 Member Function Documentation

8.45.2.1 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:

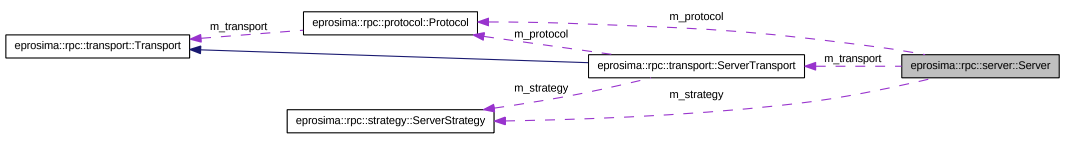
- includetmp/rpcdds/protocols/dds/MessageHeaderPlugin.h

8.46 eprosima::rpc::server::Server Class Reference

This class implements the common functionalities that any server has.

`#include <Server.h>`

Inherited by FooDDS::FooServer.Collaboration diagram for eprosima::rpc::server::Server:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_protocol
eprosima::rpc::transport::ServerTransport
m_transport
eprosima::rpc::server::Server
m_strategy
m_strategy
eprosima::rpc::strategy::ServerStrategy

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.

8.46.1 Detailed Description

This class implements the common functionalities that any server has.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 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.

8.46.3 Member Function Documentation

8.46.3.1 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:

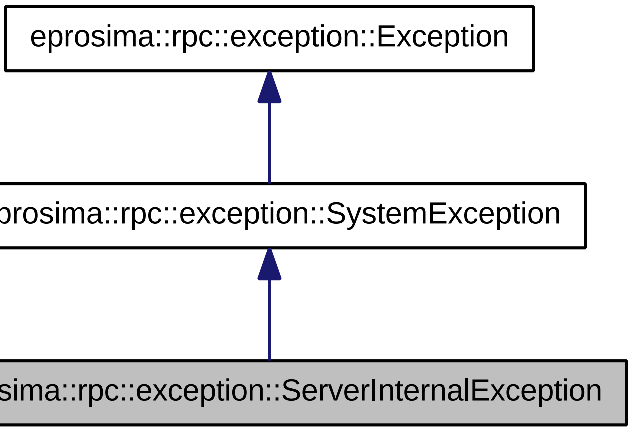
- includetmp/rpcdds/server/Server.h

8.47 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>`

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::ServerInternalException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::ServerInternalException

Public Member Functions

- ServerInternalException (const std::string &message)

  Default constructor.

- ServerInternalException (std::string &&message)

  Default constructor.

- ServerInternalException (const ServerInternalException &ex)

  Default copy constructor.

- ServerInternalException (ServerInternalException &&ex)

  Default move constructor.

- ServerInternalException & operator= (const ServerInternalException &ex)

  Assigment operation.

- ServerInternalException & operator= (ServerInternalException &&ex)

  Assigment operation.

- virtual ∼ServerInternalException () throw ()

  Default constructor.

- virtual void raise () const
    This function throws the object as an exception.

8.47.1 Detailed Description

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

8.47.2 Constructor & Destructor Documentation

8.47.2.1 eprosima::rpc::exception::ServerInternalException::ServerInternalException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.47.2.2 eprosima::rpc::exception::ServerInternalException::ServerInternalException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.47.2.3 eprosima::rpc::exception::ServerInternalException::ServerInternalException (const ServerInternalException & *ex*)

Default copy constructor.

Parameters:

ex ServerInternalException that will be copied.

8.47.2.4 eprosima::rpc::exception::ServerInternalException::ServerInternalException (ServerInternalException && *ex*)

Default move constructor.

Parameters:

ex ServerInternalException that will be moved.

8.47.3 Member Function Documentation

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

Assigment operation.

Parameters:

ex ServerInternalException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

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

Assigment operation.

Parameters:

ex ServerInternalException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

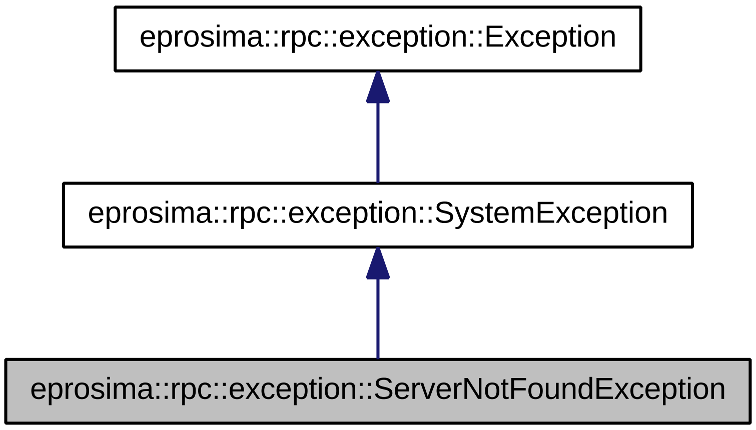
- includetmp/rpcdds/exceptions/ServerInternalException.h

8.48 eprosima::rpc::exception::ServerNotFoundException Class Reference

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

`#include <ServerNotFoundException.h>`

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::ServerNotFoundException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::ServerNotFoundException

Public Member Functions

- ServerNotFoundException (const std::string &message)
  Default constructor.
- ServerNotFoundException (std::string &&message)
  Default constructor.
- ServerNotFoundException (const ServerNotFoundException &ex)
  Default copy constructor.
- ServerNotFoundException (ServerNotFoundException &&ex)
  Default move constructor.
- ServerNotFoundException & operator= (const ServerNotFoundException &ex)
  Assigment operation.
- ServerNotFoundException & operator= (ServerNotFoundException &&ex)
  Assigment operation.
- virtual ~ServerNotFoundException () throw ()

Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.48.1 Detailed Description

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

8.48.2 Constructor & Destructor Documentation

8.48.2.1 eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.48.2.2 eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.48.2.3 eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (const ServerNotFoundException & *ex*)

Default copy constructor.

Parameters:

ex ServerNotFoundException that will be copied.

8.48.2.4 eprosima::rpc::exception::ServerNotFoundException::ServerNotFoundException (ServerNotFoundException && *ex*)

Default move constructor.

Parameters:

ex ServerNotFoundException that will be moved.

8.48.3 Member Function Documentation

8.48.3.1 ServerNotFoundException& eprosima::rpc::exception::ServerNotFoundException::operator= (ServerNotFoundException && *ex*)

Assigment operation.

Parameters:

ex ServerNotFoundException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

8.48.3.2 ServerNotFoundException& eprosima::rpc::exception::ServerNotFoundException::operator= (const ServerNotFoundException & *ex*)

Assigment operation.

Parameters:

ex ServerNotFoundException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

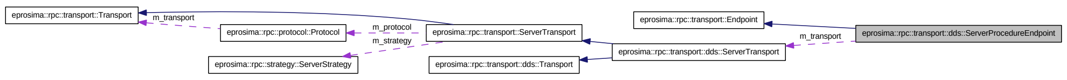
- includetmp/rpcdds/exceptions/ServerNotFoundException.h

8.49 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>`

Inherits eprosima::rpc::transport::Endpoint.Collaboration diagram for eprosima::rpc::transport::dds::ServerProcedureEndpoint:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_strategy
eprosima::rpc::transport::ServerTransport
eprosima::rpc::strategy::ServerStrategy
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::Endpoint
eprosima::rpc::transport::dds::ServerTransport
m_transport
eprosima::rpc::transport::dds::ServerProcedureEndpoint

Public Member Functions

- ServerProcedureEndpoint (ServerTransport &transport)

  Default constructor.

- virtual ∼ServerProcedureEndpoint ()

  Default destructor.

- 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.

- int start (std::string &serviceName)

  This method creates the DDS entities needed to run this DDS Endpoint.

- void stop ()

  This method deletes the DDS entities needed to run this DDS Endpoint.

- Transport::ProcessFunc getProcessFunc ()

  Gets the callback used to processes a request.

- int sendReply (void ∗data)

  Sends the reply.

- virtual void on_data_available (DDS::DataReader ∗reader)

  DDS callback.

- virtual void on_requested_deadline_missed (DDS::DataReader ∗reader, const DDS::RequestedDeadlineMissedStatus &status)

  DDS callback.

- virtual void on_requested_incompatible_qos (DDS::DataReader ∗reader, const DDS::RequestedIncompatibleQosStatus &status)

  DDS callback.

- virtual void on_sample_rejected (DDS::DataReader ∗reader, const DDS::SampleRejectedStatus &status)

  DDS callback.

- virtual void on_liveliness_changed (DDS::DataReader ∗reader, const DDS::LivelinessChangedStatus &status)

  DDS callback.

- virtual void on_sample_lost (DDS::DataReader ∗reader, const DDS::SampleLostStatus &status)

  DDS callback.

- virtual void on_subscription_matched (DDS::DataReader ∗reader, const DDS::SubscriptionMatchedStatus &status)

  DDS callback.

8.49.1 Detailed Description

This class represents a remote endpoint used by a proxy. Also this class encapsulate the DDS datawriter and the DDS datareader.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 eprosima::rpc::transport::dds::ServerProcedureEndpoint::ServerProcedureEndpoint (ServerTransport & *transport*)

Default constructor.

Parameters:

Transport that creates the proxy procedure endpoint. It cannot be NULL.

8.49.3 Member Function Documentation

8.49.3.1 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.

8.49.3.2 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

8.49.3.3 int eprosima::rpc::transport::dds::ServerProcedureEndpoint::sendReply (void ∗ *data*)

Sends the reply.

Parameters:

serviceName Name of the service.

8.49.3.4 int eprosima::rpc::transport::dds::ServerProcedureEndpoint::start (std::string & *serviceName*)

This method creates the DDS entities needed to run this DDS Endpoint.

Parameters:

serviceName Name of the service.

8.49.3.5 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

8.50 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>
```

Inherited by eprosima::rpc::strategy::SingleThreadStrategy, eprosima::rpc::strategy::ThreadPerRequestStrategy, and eprosima::rpc::strategy::ThreadPoolStrategy.

Public Member Functions

- ServerStrategy ()
  Default constructor.
- virtual ∼ServerStrategy ()
  Default destructor.
- virtual ServerStrategyImpl ∗ getImpl ()=0
  Gets the implementation of the strategy using Boost library.

8.50.1 Detailed Description

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

8.50.2 Member Function Documentation

8.50.2.1 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::SingleThreadStrategy, eprosima::rpc::strategy::ThreadPerRequestStrategy, and eprosima::rpc::strategy::ThreadPoolStrategy.

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

- includetmp/rpcdds/strategies/ServerStrategy.h

8.51 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.

8.51.1 Detailed Description

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

8.51.2 Member Function Documentation

8.51.2.1 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:

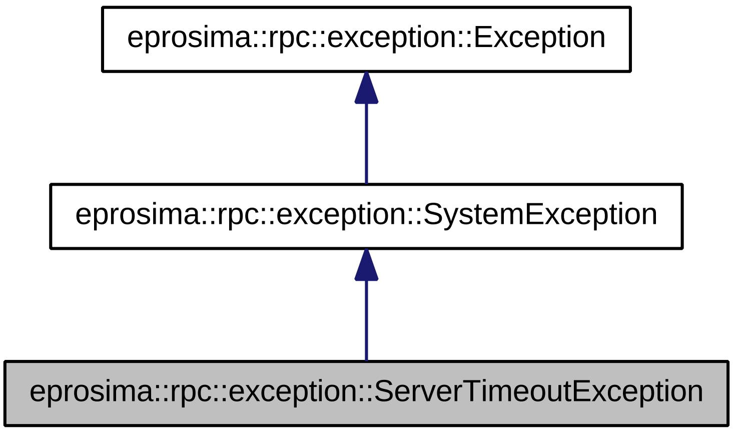
- includetmp/rpcdds/strategies/ServerStrategyImpl.h

8.52 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>
```

Inherits eprosima::rpc::exception::SystemException.Collaboration diagram for eprosima::rpc::exception::ServerTimeoutException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException
eprosima::rpc::exception::ServerTimeoutException

Public Member Functions

- ServerTimeoutException (const std::string &message)

  Default constructor.
- ServerTimeoutException (std::string &&message)

  Default constructor.
- ServerTimeoutException (const ServerTimeoutException &ex)

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

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

  Assigment operation.
- ServerTimeoutException & operator= (ServerTimeoutException &&ex)

  Assigment operation.
- virtual ∼ServerTimeoutException () throw ()

Default constructor.

- virtual void raise () const

  This function throws the object as an exception.

8.52.1 Detailed Description

This class is thrown as an exception when the remote procedure call exceeds the maximum time.

8.52.2 Constructor & Destructor Documentation

8.52.2.1 eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (const std::string & *message*)

Default constructor.

Parameters:

message An error message. This message is copied.

8.52.2.2 eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (std::string && *message*)

Default constructor.

Parameters:

message An error message. This message is moved.

8.52.2.3 eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (const ServerTimeoutException & *ex*)

Default copy constructor.

Parameters:

ex ServerTimeoutException that will be copied.

8.52.2.4 eprosima::rpc::exception::ServerTimeoutException::ServerTimeoutException (ServerTimeoutException && *ex*)

Default move constructor.

Parameters:

ex ServerTimeoutException that will be moved.

8.52.3 Member Function Documentation

8.52.3.1 ServerTimeoutException& eprosima::rpc::exception::ServerTimeoutException::operator= (ServerTimeoutException && *ex*)

Assigment operation.

Parameters:

ex ServerTimeoutException that will be moved.

Reimplemented from eprosima::rpc::exception::SystemException.

8.52.3.2 ServerTimeoutException& eprosima::rpc::exception::ServerTimeoutException::operator= (const ServerTimeoutException & *ex*)

Assigment operation.

Parameters:

ex ServerTimeoutException that will be copied.

Reimplemented from eprosima::rpc::exception::SystemException.

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

- includetmp/rpcdds/exceptions/ServerTimeoutException.h

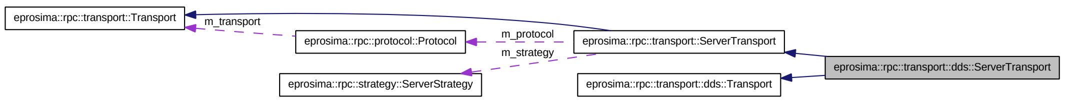
8.53 eprosima::rpc::transport::dds::ServerTransport Class Reference

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>`

Inherits eprosima::rpc::transport::ServerTransport, and eprosima::rpc::transport::dds::Transport.

Inherited by eprosima::rpc::transport::dds::TCPServerTransport, and eprosima::rpc::transport::dds::UDPServerTransport.Collaboration diagram for eprosima::rpc::transport::dds::ServerTransport:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_strategy
eprosima::rpc::transport::ServerTransport
eprosima::rpc::strategy::ServerStrategy
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::dds::ServerTransport

Public Member Functions

- virtual ~ServerTransport ()

  Default destructor.

- virtual const char ∗ getType () const

  This function returns the type of the transport. This function has to be implemented by the child classes.

- 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.

- void process (ServerProcedureEndpoint ∗endpoint, void ∗data)

  This method is invoked once for each incoming request.

- void run ()

  This method starts all the DDS Datawriters and Datareaders.

- 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.

- int receive (char ∗buffer, size_t bufferLength, size_t &dataToRead, Endpoint ∗endpoint)

  This function does not apply to DDS transport.

Protected Member Functions

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)=0

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

- ServerTransport (std::string &serviceName, int domainId=0)

  Default constructor.

8.53.1 Detailed Description

This class is the base of all classes that implement a transport using DDS. This transport can be used by the servers.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 eprosima::rpc::transport::dds::ServerTransport::ServerTransport (std::string & *serviceName*, int *domainId* = `0`) `[protected]`

Default constructor.

Parameters:

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

8.53.3 Member Function Documentation

8.53.3.1 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.

8.53.3.2 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.

8.53.3.3 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.

8.53.3.4 virtual int eprosima::rpc::transport::dds::ServerTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *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

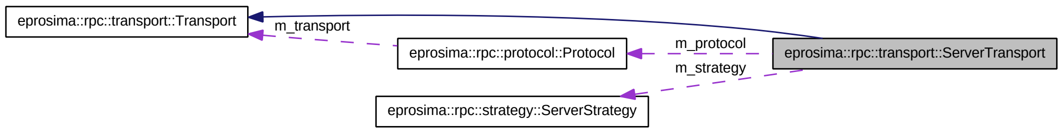
8.54 eprosima::rpc::transport::ServerTransport Class Reference

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

```
#include <ServerTransport.h>
```

Inherits eprosima::rpc::transport::Transport.

Inherited by eprosima::rpc::transport::dds::ServerTransport.Collaboration diagram for eprosima::rpc::transport::ServerTransport:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_strategy
eprosima::rpc::transport::ServerTransport
eprosima::rpc::strategy::ServerStrategy

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.

8.54.1 Detailed Description

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

8.54.2 Member Function Documentation

8.54.2.1 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.

8.54.2.2 ServerTransport_Callback eprosima::rpc::transport::ServerTransport::getCallback () const `[inline]`

Gets the callback that will proccess the requests.

Returns:

Callback that will proccess the requests.

8.54.2.3 eprosima::rpc::protocol::Protocol& eprosima::rpc::transport::ServerTransport::getLinkedProtocol () `[inline]`

Gets the communication protocol.

Returns:

Communication protocol.

8.54.2.4 eprosima::rpc::strategy::ServerStrategy& eprosima::rpc::transport::ServerTransport::getStrategy () const `[inline]`

Gets the threading strategy.

Returns:

Threading strategy.

8.54.2.5 void eprosima::rpc::transport::ServerTransport::linkProtocol (eprosima::rpc::protocol::Protocol & *protocol*) `[inline]`

Sets the communication protocol.

Parameters:

protocol Communication protocol.

8.54.2.6 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.

8.54.2.7 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.

8.54.2.8 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.

8.54.2.9 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:

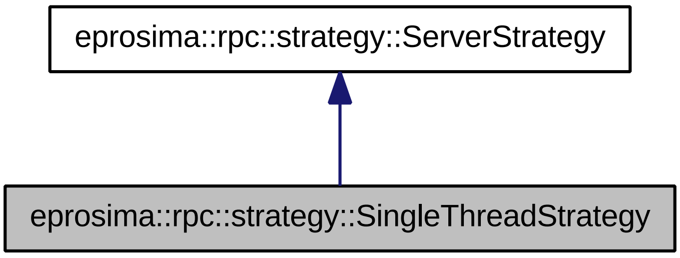
- includetmp/rpcdds/transports/ServerTransport.h

8.55 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>`

Inherits eprosima::rpc::strategy::ServerStrategy.Collaboration diagram for eprosima::rpc::strategy::SingleThreadStrategy:


eprosima::rpc::strategy::ServerStrategy
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.

8.55.1 Detailed Description

This class implements the sigle thread strategy. The server uses a reception thread to execute all the requests.

8.55.2 Member Function Documentation

8.55.2.1 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

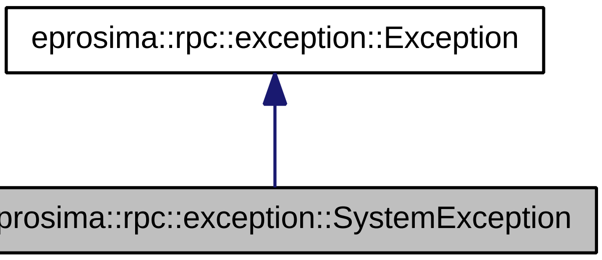
8.56 eprosima::rpc::exception::SystemException Class Reference

This abstract class is used to create internal FASTRPC exceptions.

```
#include <SystemException.h>
```

Inherits eprosima::rpc::exception::Exception.

Inherited by eprosima::rpc::exception::BadParamException, eprosima::rpc::exception::ClientInternalException, eprosima::rpc::exception::IncompatibleException, eprosima::rpc::exception::InitializeException, eprosima::rpc::exception::ServerInternalException, eprosima::rpc::exception::ServerNotFoundException, and eprosima::rpc::exception::ServerTimeoutException.Collaboration diagram for eprosima::rpc::exception::SystemException:


eprosima::rpc::exception::Exception
eprosima::rpc::exception::SystemException

Public Member Functions

- virtual ~SystemException () throw ()

  Default destructor.

- const int32_t & minor () const

  This function returns the number associated with the system exception.

- void minor (const int32_t &minor)

  This function sets the number that will be associated with the system exception.

- virtual void raise () const =0

  This function throws the object as an exception.

- virtual const char ∗ what () const throw ()

  This function returns the error message.

Protected Member Functions

- SystemException (const std::string &message)

Default constructor.

- SystemException (std::string &&message)

  Default constructor.

- SystemException (const SystemException &ex)

  Default copy constructor.

- SystemException (SystemException &&ex)

  Default move constructor.

- SystemException (const std::string &message, int32_t minor)

  Constructor.

- SystemException (std::string &&message, int32_t minor)

  Constructor.

- SystemException & operator= (const SystemException &ex)

  Assigment operation.

- SystemException & operator= (SystemException &&ex)

  Assigment operation.

8.56.1 Detailed Description

This abstract class is used to create internal FASTRPC exceptions.

8.56.2 Constructor & Destructor Documentation

8.56.2.1 eprosima::rpc::exception::SystemException::SystemException (const std::string & *message*) `[protected]`

Default constructor.

Parameters:

message An error message. This message is copied.

8.56.2.2 eprosima::rpc::exception::SystemException::SystemException (std::string && *message*) `[protected]`

Default constructor.

Parameters:

message An error message. This message is moved.

8.56.2.3 eprosima::rpc::exception::SystemException::SystemException (const SystemException & *ex*) `[protected]`

Default copy constructor.

Parameters:

ex SystemException that will be copied.

8.56.2.4 eprosima::rpc::exception::SystemException::SystemException (SystemException && *ex*) `[protected]`

Default move constructor.

Parameters:

ex SystemException that will be moved.

8.56.2.5 eprosima::rpc::exception::SystemException::SystemException (const std::string & *message*, int32_t *minor*) `[protected]`

Constructor.

Parameters:

message An error message. This message is copied.

minor The number that will be associated with the system exception.

8.56.2.6 eprosima::rpc::exception::SystemException::SystemException (std::string && *message*, int32_t *minor*) `[protected]`

Constructor.

Parameters:

message An error message. This message is moved.

minor The number that will be associated with the system exception.

8.56.3 Member Function Documentation

8.56.3.1 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.

8.56.3.2 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.

8.56.3.3 SystemException& eprosima::rpc::exception::SystemException::operator= (SystemException && *ex*) `[protected]`

Assigment operation.

Parameters:

ex SystemException that will be moved.

Reimplemented from eprosima::rpc::exception::Exception.

Reimplemented in eprosima::rpc::exception::BadParamException, eprosima::rpc::exception::ClientInternalException, eprosima::rpc::exception::IncompatibleException, eprosima::rpc::exception::InitializeException, eprosima::rpc::exception::ServerInternalException, eprosima::rpc::exception::ServerNotFoundException, and eprosima::rpc::exception::ServerTimeoutException.

8.56.3.4 SystemException& eprosima::rpc::exception::SystemException::operator= (const SystemException & *ex*) `[protected]`

Assigment operation.

Parameters:

ex SystemException that will be copied.

Reimplemented from eprosima::rpc::exception::Exception.

Reimplemented in eprosima::rpc::exception::BadParamException, eprosima::rpc::exception::ClientInternalException, eprosima::rpc::exception::IncompatibleException, eprosima::rpc::exception::InitializeException, eprosima::rpc::exception::ServerInternalException, eprosima::rpc::exception::ServerNotFoundException, and eprosima::rpc::exception::ServerTimeoutException.

8.56.3.5 virtual 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:

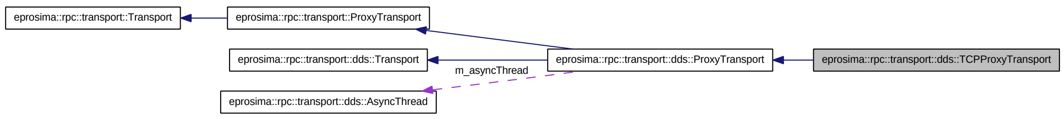
- includetmp/rpcdds/exceptions/SystemException.h

8.57 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>`

Inherits eprosima::rpc::transport::dds::ProxyTransport.Collaboration diagram for eprosima::rpc::transport::dds::TCPProxyTransport:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::dds::Transport
m_asyncThread
eprosima::rpc::transport::dds::ProxyTransport
eprosima::rpc::transport::dds::TCPProxyTransport
eprosima::rpc::transport::dds::AsyncThread

Public Member Functions

- TCPProxyTransport (const char ∗to_connect, std::string remoteServiceName, int domainId=0, long timeout=10000L)

  Default constructor for the proxies.

- virtual ∼TCPProxyTransport ()

  Default destructor.

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)

  This function sets the DDS' QoS to use the TCPv4 transport.

8.57.1 Detailed Description

This class implements a transport using DDS over TCPv4. This transport can only be used by a server proxy.

8.57.2 Constructor & Destructor Documentation

8.57.2.1 eprosima::rpc::transport::dds::TCPProxyTransport::TCPProxyTransport (const char ∗ *to_connect*, std::string *remoteServiceName*, 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"

remoteServiceName 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.

8.57.3 Member Function Documentation

8.57.3.1 virtual 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.

Implements eprosima::rpc::transport::dds::ProxyTransport.

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

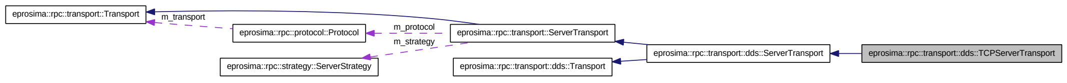
- includetmp/rpcdds/transports/dds/TCPProxyTransport.h

8.58 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>`

Inherits eprosima::rpc::transport::dds::ServerTransport.Collaboration diagram for eprosima::rpc::transport::dds::TCPServerTransport:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_strategy
eprosima::rpc::transport::ServerTransport
eprosima::rpc::strategy::ServerStrategy
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::dds::ServerTransport
eprosima::rpc::transport::dds::TCPServerTransport

Public Member Functions

- TCPServerTransport (const char ∗public_address, const char ∗server_bind_port, std::string serviceName, int domainId=0)

  Default constructor for servers.

- virtual ∼TCPServerTransport ()

  Default destructor.

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)

  This function sets the QoS to use the TCPv4 transport.

8.58.1 Detailed Description

This class implements a transport using DDS over TCPv4. This transport can only be used by a server.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 eprosima::rpc::transport::dds::TCPServerTransport::TCPServerTransport (const char ∗ *public_address*, const char ∗ *server_bind_port*, std::string *serviceName*, int *domainId* = 0)

Default constructor for servers.

Parameters:

public_address Public address and port of the server. The server should be accessible 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.

remoteServiceName Name of the remote service

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

8.58.3 Member Function Documentation

8.58.3.1 virtual int eprosima::rpc::transport::dds::TCPServerTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *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:

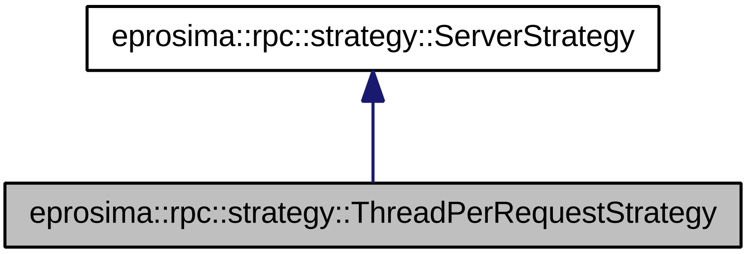
- includetmp/rpcdds/transports/dds/TCPServerTransport.h

8.59 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>`

Inherits eprosima::rpc::strategy::ServerStrategy.Collaboration diagram for eprosima::rpc::strategy::ThreadPerRequestStrategy:


eprosima::rpc::strategy::ServerStrategy
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.

8.59.1 Detailed Description

This class implements the thread per request strategy. The server creates a new thread for every new incoming request.

8.59.2 Member Function Documentation

8.59.2.1 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:

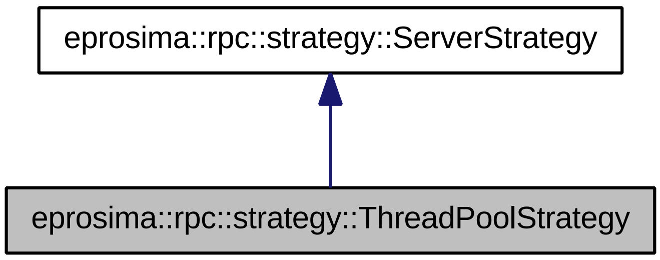
- includetmp/rpcdds/strategies/ThreadPerRequestStrategy.h

8.60 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>`

Inherits eprosima::rpc::strategy::ServerStrategy.Collaboration diagram for eprosima::rpc::strategy::ThreadPoolStrategy:


eprosima::rpc::strategy::ServerStrategy
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.

8.60.1 Detailed Description

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

8.60.2 Constructor & Destructor Documentation

8.60.2.1 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.

8.60.3 Member Function Documentation

8.60.3.1 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

8.61 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>`

Inherited by eprosima::rpc::transport::ProxyTransport, and eprosima::rpc::transport::ServerTransport.

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

8.61.1 Detailed Description

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

8.61.2 Member Function Documentation

8.61.2.1 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::ProxyTransport, and eprosima::rpc::transport::ServerTransport.

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

- includetmp/rpcdds/transports/Transport.h

8.62 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>`

Inherited by eprosima::rpc::transport::dds::ProxyTransport, and eprosima::rpc::transport::dds::ServerTransport.

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.

- DDS::DomainParticipant ∗ getParticipant () const

  Gets the domain participant.

- DDS::Publisher ∗ getPublisher () const

  Gets the publisher.

- DDS::Subscriber ∗ 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, DDS::DomainParticipant ∗participant)=0

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

- Transport (int domainId=0)

  Default constructor.

8.62.1 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.

8.62.2 Constructor & Destructor Documentation

8.62.2.1 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.

8.62.3 Member Function Documentation

8.62.3.1 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.

8.62.3.2 DDS::DomainParticipant∗ eprosima::rpc::transport::dds::Transport::getParticipant () const `[inline]`

Gets the domain participant.

Returns:

DDS domain participant.

8.62.3.3 DDS::Publisher∗ eprosima::rpc::transport::dds::Transport::getPublisher () const `[inline]`

Gets the publisher.

Returns:

DDS publisher.

8.62.3.4 DDS::Subscriber∗ eprosima::rpc::transport::dds::Transport::getSubscriber () const `[inline]`

Gets the subscriber.

Returns:

DDS subscriber.

8.62.3.5 virtual int eprosima::rpc::transport::dds::Transport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *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::TCPProxyTransport, eprosima::rpc::transport::dds::TCPServerTransport, eprosima::rpc::transport::dds::UDPProxyTransport, and eprosima::rpc::transport::dds::UDPServerTransport.

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

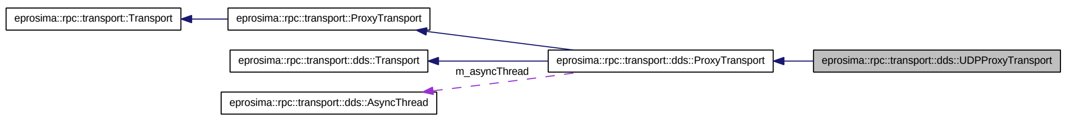
- includetmp/rpcdds/transports/dds/Transport.h

8.63 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>
```

Inherits eprosima::rpc::transport::dds::ProxyTransport.Collaboration diagram for eprosima::rpc::transport::dds::UDPProxyTransport:


eprosima::rpc::transport::Transport
eprosima::rpc::transport::ProxyTransport
eprosima::rpc::transport::dds::Transport
m_asyncThread
eprosima::rpc::transport::dds::ProxyTransport
eprosima::rpc::transport::dds::UDPProxyTransport
eprosima::rpc::transport::dds::AsyncThread

Public Member Functions

- UDPProxyTransport (std::string remoteServiceName, int domainId=0, long timeout=10000L)

  Default constructor for server's proxies.

- UDPProxyTransport (const char ∗to_connect, std::string remoteServiceName, int domainId=0, long timeout=10000L)

  Constructor for server's proxies.

- virtual ∼UDPProxyTransport ()

  Default destructor.

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)

  This function sets the QoS of DDS to use the UDPv4 transport.

8.63.1 Detailed Description

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

8.63.2 Constructor & Destructor Documentation

8.63.2.1 eprosima::rpc::transport::dds::UDPProxyTransport::UDPProxyTransport (std::string *remoteServiceName*, 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.

8.63.2.2 eprosima::rpc::transport::dds::UDPProxyTransport::UDPProxyTransport (const char * *to_connect*, std::string *remoteServiceName*, 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.

8.63.3 Member Function Documentation

8.63.3.1 virtual int eprosima::rpc::transport::dds::UDPProxyTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant * *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:

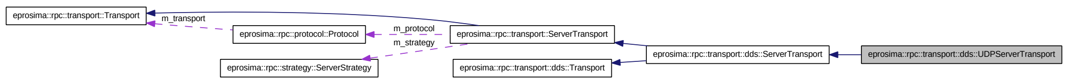
- includetmp/rpcdds/transports/dds/UDPProxyTransport.h

8.64 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>`

Inherits eprosima::rpc::transport::dds::ServerTransport.Collaboration diagram for eprosima::rpc::transport::dds::UDPServerTransport:


eprosima::rpc::transport::Transport
m_transport
eprosima::rpc::protocol::Protocol
m_protocol
m_strategy
eprosima::rpc::transport::ServerTransport
eprosima::rpc::strategy::ServerStrategy
eprosima::rpc::transport::dds::Transport
eprosima::rpc::transport::dds::ServerTransport
eprosima::rpc::transport::dds::UDPServerTransport

Public Member Functions

- UDPServerTransport (std::string serviceName, int domainId=0)

  Default constructor for servers.

- virtual ∼UDPServerTransport ()

  Default destructor.

- virtual int setTransport (DDS::DomainParticipantQos &participantQos, DDS::DomainParticipant ∗participant)

  This function sets the DDS' QoS to use the UDPv4 transport.

8.64.1 Detailed Description

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

8.64.2 Constructor & Destructor Documentation

8.64.2.1 eprosima::rpc::transport::dds::UDPServerTransport::UDPServerTransport (std::string *serviceName*, 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.

8.64.3 Member Function Documentation

8.64.3.1 virtual int eprosima::rpc::transport::dds::UDPServerTransport::setTransport (DDS::DomainParticipantQos & *participantQos*, DDS::DomainParticipant ∗ *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:

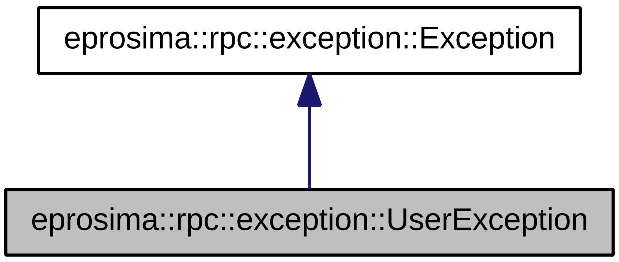
- includetmp/rpcdds/transports/dds/UDPServerTransport.h

8.65 eprosima::rpc::exception::UserException Class Reference

This abstract class is used to create user exceptions.

`#include <UserException.h>`

Inherits eprosima::rpc::exception::Exception.Collaboration diagram for eprosima::rpc::exception::UserException:


eprosima::rpc::exception::Exception
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.

8.65.1 Detailed Description

This abstract class is used to create user exceptions.

8.65.2 Constructor & Destructor Documentation

8.65.2.1 eprosima::rpc::exception::UserException::UserException (const UserException & *ex*) `[protected]`

Default copy constructor.

Parameters:

ex UserException that will be copied.

8.65.2.2 eprosima::rpc::exception::UserException::UserException (UserException && *ex*) `[protected]`

Default move constructor.

Parameters:

ex UserException that will be moved.

8.65.3 Member Function Documentation

8.65.3.1 UserException& eprosima::rpc::exception::UserException::operator= (UserException && *ex*) `[protected]`

Assigment operation.

Parameters:

ex UserException that will be moved.

Reimplemented from eprosima::rpc::exception::Exception.

8.65.3.2 UserException& eprosima::rpc::exception::UserException::operator= (const UserException & *ex*) `[protected]`

Assigment operation.

Parameters:

ex UserException that will be copied.

Reimplemented from eprosima::rpc::exception::Exception.

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

- includetmp/rpcdds/exceptions/UserException.h

Index