KnoRBA C++ Library

v0.6

Generated by Doxygen 1.8.4

Fri Oct 30 2015 13:07:18

Contents

Chapter 1

Overview

Creating Agents

Documentation for knorba::Agent contains basic instruction for creating agents.

Code reusability in KnoRBA agent-based component model is different than object-oriented paradigm, in sense that it is horizontal rather than vertical. Reusable modules are composed into protocols, which can be included by multiple agents. See documentation for knorba::Protocol for more details.

Type Wrapper Classes

Since KnoRBA data types used for sending and receiving messages are different than native C++ types, there are a rich set of type-wrapper classes provided to easily create and manipulate values in portable KnoRBA types. The following table summerizes these classes.

KnoRBA Type	**Encoding**	**Type Info**	**Wrapper Class**	**Scalar Type**
octet	8-bit unsigned integer	KOctet	KType::OCTET	k_octet_t
integer	32-bit 2's complement signed integer	KType::INTEGER	KInteger	k_integer_t
longint	64-bit 2's complement signed integer	KType::LONGINT	KLongint	k_longint_t
real	64-bit IEEE 754 floating point	KType::REAL	KReal	k_real_t
guid	128-bit globally unique ID	KType::GUID	KGuid	k_guid_t
string	UTF-8	KType::STRING	KString	-
raw	octets	KType::RAW	KRaw	-
enumeration	1 octet	KEnumerationType	KEnumeration	-
record		KRecordType	KRecord	-
grid		KGridType	KGrid	-
any		KType::ANY	KAny	-
nothing		KType::NOTHING	KValue::NOTHING	-

Brows the list of classes for details on above items.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 knorba::Agent Class Reference

Extend this class to implement a KnoRBA agent.

`#include <knorba/Agent.h>`

Public Types

- typedef void(Agent::∗ handler_t)(**PPtr**< Message >)

  Pointer to handler method.

Public Member Functions

- Agent (Runtime &rt, const k_guid_t &guid, int queueSize=DEFAULT_QUEUE_SIZE)

  Sole constructor.
- virtual ∼Agent ()

  Deconstructor.
- void run ()

  FOR INTERNAL USE.
- void quit ()

  Runs the finalizer thread, which stops the message processor thread and runs the finalize() method.
- bool isPassive () const

  Checks if this agent is marked as passive.
- void registerProtocol (Protocol ∗protocol)

  FOR INTERNAL USE.
- void unregisterProtocol (Protocol ∗protocol)

  FOR INTERNAL USE.
- void addPeer (**PPtr**< KString > role, const k_guid_t &guid)

  Adds a peer with the given GUID to the given role.
- void removePeer (**PPtr**< KString > role, const k_guid_t &guid)

  Removes the peer with the given GUID from the given role.
- void removeAllPeers (**PPtr**< KString > role)

  Removes all peers associated with the given role.
- void removeAllPeersWithMatchingAppId (const k_guid_t &guid)

  Removes all peers that share the same AppId as in the given GUID.
- bool isPeer (const k_guid_t &guid) const

Checks whether or not the given GUID belongs to a registered peer.

- **PPtr**< KString > getRole (const k_guid_t &guid) const

  Returns the role of the peer with the given GUID.
- **PPtr**< Group > getPeers (**PPtr**< KString > role) const

  Returns all the peers associated with the given role.
- **PPtr**< Group > getAllPeers () const

  Returns a group of all the registered peers.
- void send (const k_guid_t receiver, **PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a message to another agent.
- void send (**PPtr**< Group > receivers, **PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a multicast message to a group of agents.
- void send (**PPtr**< KString > role, **PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a multicast message to all agents with the given role.
- void sendToAll (**PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a broadcast message.
- void sendToLocals (**PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a broadcast message to all local agents.
- void respond (**PPtr**< Message > msg, **PPtr**< KString > opcode, **PPtr**< KValue > content)

  Sends a message in response to a received message.
- **Ptr**< Message > tsend (const k_guid_t receiver, **PPtr**< KString > opcode, **PPtr**< KValue > content, k_-integer_t timeout=-1)

  Blocking unicast send.
- **Ptr**< MessageSet > tsend (**PPtr**< Group > receivers, **PPtr**< KString > opcode, **PPtr**< KValue > content, k_integer_t timeout=-1)

  Blocking multicast send.
- **Ptr**< MessageSet > tsend (**PPtr**< KString > receivers, **PPtr**< KString > opcode, **PPtr**< KValue > content, k_integer_t timeout=-1)

  Blocking multicast send to peers.
- **Ptr**< MessageSet > tsendToLocals (**PPtr**< KString > opcode, **PPtr**< KValue > content, k_integer_t time-out)

  Blocking local broadcast.
- const k_guid_t & getGuid () const

  Returns the GUID of this agent.
- **Logger::Stream** & log (const **Logger::level_t** level=**Logger::L3**) const

  Returns a logger stream into the default logger, beginning with the identity of this agent.
- **PPtr**< **Path** > getPathToResources () const

  Return the path to resouces for this agent.
- **PPtr**< **Path** > getPathToData () const

  Returns the path to the folder in which this agent can store its data.
- const string & getAlias () const

  Return the alias for this agent.
- Runtime & getRuntime ()

  Returns reference to runtime interface.
- virtual void handlePeerConnectionRequest (**PPtr**< KString > role, const k_guid_t &guid)

  Override to handle peer connection request.
- virtual void handlePeerDisconnected (**PPtr**< KString > role, const k_guid_t &guid)

  Override to handle peer disconnect notifications.
- virtual bool isAlive ()

  Returns true if the agent is alive.
- virtual void finalize ()

  Override to perform additional tasks when agent is finalizing.

Static Public Attributes

- static const int DEFAULT_QUEUE_SIZE = 16

    Default queue size.
- static const **SPtr**< KString > OP_CONNECT = **Ptr**<KString>(new KString("knorba.agent.connect"))

    Opcode for connect request message.
- static const **SPtr**< KString > OP_ACK = **Ptr**<KString>(new KString("knorba.agent.ack"))

    Opcode for acknowledge [response] message.
- static const **SPtr**< KString > OP_NG = **Ptr**<KString>(new KString("knorba.agent.ng"))

    Opcode for NG message [response] message.

Protected Member Functions

- void setPassive (bool value=true)

    Marks this agent as passive.
- void sleep (int msecs)

    Pauses the current thread while making sure the message processor thread is always running.
- void registerHandler (handler_t h, const **PPtr**< KString > opcode)

    Registers a handler for the given opcode.

4.1.1 Detailed Description

Extend this class to implement a KnoRBA agent.

A KnoRBA app terminates only if every nonpassive agents (see below) call quit(). Make sure to override isAlive() and finalize() methods if your agent creates any user threads. Use Protocol class to implement behaviors shared between various types of agents.

Sending and Receiving Messages

The basic implementation of an agent involves implementing a set of message handlers with

```
void MyAgent::handlerName(PPtr<Message> msg)
```

signiture.

Each handler should be explicitly registered to work. This is usually done in the constructor.

```
MyAgent::MyAgent(Runtime& rt, k_guid_t& guid)
: Agent(rt, guid)
{
    registerHandler((handler_t)&MyAgent::handlerName, OP_CODE);
}
```

OP_CODE shoule be a Ptr<KString>. After registered, the handler will be called any time the agents receives a message with the given opcode.

Incomming messages are queued and processed sequentially. That means, first, no two handlers can manipulate the same data at the same time. But if also means that if a handler takes too much time to process a message, it may cause congestion and eventually overload in the message queue. However, you may use blocking tsendXXX methods safely as they internally assure continues execution of message thread, while waiting. Use Agent::sleep() instead of **System::sleep()** or std::sleep().

To communicate with other agents, use sendXXX and tsendXXX methods. Because of asynchronous nature of KnoRBA, primitive send operations are non-blocking. However, you have the option to block the sender agent until the remote agent reponds by using tsendXXX. These methods create a transaction and keep it open until all target remote agents respond.

The receving agent can check if it is at the receiving end of a transaction by invocking Message::needsResponse(), and if it is respond using Agent::respond() method.

Peer Management

Each agent can define a set of roles, and each role can be fulfilled by a set of other agents, known as peers.

In KnoRBA peers may disappear unexpectly, or appear at any time. Override handlePeerConnectionRequest() and handlePeerDisconnected() to react to changes as appropriate.

Use the following methods to mamange peers: addPeer(), removePeer(), removeAllPeers(), removeAllPeersWith-MatchingAppId(), isPeer(), getRole(), getPeers(), and getAllPeers().

Passive Agents

The sole condition for a KnoRBA app to terminate is all agents in that app to terminate. Normal agents terminate only in two ways, either by calling quit() method volunteerly, or when the system is shutting down. Passive agents, on the other hand, will automatically quit when all other nonpassive agents quit. To define make an agent passive, call setPassive() in the constructor.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 knorba::Agent::Agent (**Runtime** & *rt,* const **k_guid_t** & *guid,* int *queueSize* = **DEFAULT_QUEUE_SIZE**)

Sole constructor.

Reference to runtime and GUID are provided by runtime when initializing dynamic library containing the agent code.

Parameters

rt	Reference to runtime access API
guid	The GUID allocated by runtime for this agent.
queueSize	The maximum number of message can be kept in the queue at any given time. Default value is 16.

4.1.2.2 knorba::Agent::∼Agent ( ) [virtual]

Deconstructor.

It advisable to override finalize() method instead of overriding the deconstructor.

4.1.3 Member Function Documentation

4.1.3.1 void knorba::Agent::addPeer (**PPtr**< **KString** > *role,* const **k_guid_t** & *guid*)

Adds a peer with the given GUID to the given role.

Parameters

role	The role for the peer to be added
guid	The GUID of the peer to be added.

4.1.3.2 void knorba::Agent::finalize ( ) [virtual]

Override to perform additional tasks when agent is finalizing.

Stops the message processor thread.

See Also

isAlive()
Protocol::finalize()

4.1.3.3 PPtr< Group > knorba::Agent::getPeers (PPtr< KString > *role*) const

Returns all the peers associated with the given role.

Returns an empty group of the given role does not exist.

4.1.3.4 PPtr< KString > knorba::Agent::getRole (const k_guid_t & *guid*) const

Returns the role of the peer with the given GUID.

Returns NULL if the given GUID does not belong to a peer.

4.1.3.5 void knorba::Agent::handlePeerConnectionRequest (PPtr< KString > *role*, const k_guid_t & *guid*) [virtual]

Override to handle peer connection request.

Default behavior is to forward the request to all protocols, if any.

Parameters

role	The request role for the new peer.
guid	The GUID of the agent requesting to become a peer.

See Also

handlePeerDisconnected()
Protocol::handlePeerConnectionRequest

4.1.3.6 void knorba::Agent::handlePeerDisconnected (PPtr< KString > *role*, const k_guid_t & *guid*) [virtual]

Override to handle peer disconnect notifications.

Default behavior is to forward the request to all protocols, if any.

Parameters

role	The role of the peer to be removed.
guid	The GUID of the agent requesting to be removed as peer.

See Also

handlePeerConnectionRequest()
Protocol::handlePeerDisconnected()

4.1.3.7 bool knorba::Agent::isAlive ( ) [virtual]

Returns true if the agent is alive.

As long as this method returns true, the ARE containing this agent will not shut down. Override if there are additional criteria to determine this agent is alive. E.g. other threads are running, connections are open, etc.

See Also

finalize()
Protocol::isAlive()

4.1.3.8 Logger::Stream & knorba::Agent::log (const Logger::level_t *level* = Logger::L3) const

Returns a logger stream into the default logger, beginning with the identity of this agent.

Usage:

this->log() << "Hello!" << EL;

Parameters

level	The log level. Default value is **Logger::L3**.

4.1.3.9 void knorba::Agent::quit ( )

Runs the finalizer thread, which stops the message processor thread and runs the finalize() method.

If successful, informs the runtime, which will release resources consumed by this agent.

4.1.3.10 void knorba::Agent::registerHandler (handler_t *h*, const PPtr< KString > *opcode*) [protected]

Registers a handler for the given opcode.

Parameters

h	Pointer to handler method
opcode	The opcode that activates the given handler

4.1.3.11 void knorba::Agent::registerProtocol (Protocol * *protocol*)

FOR INTERNAL USE.

Do not call directly. Activates support for the given protocol in this agent.

4.1.3.12 void knorba::Agent::removeAllPeers (PPtr< KString > *role*)

Removes all peers associated with the given role.

If the indicated role does not exist, the method will end successfully without any effects.

Parameters

role	The role to be removed.

4.1.3.13 void knorba::Agent::removeAllPeersWithMatchingAppId (const k_guid_t & *guid*)

Removes all peers that share the same AppId as in the given GUID.

Parameters

guid	The AppID part of this GUID will be matched against all peers of this agent.

4.1.3.14 void knorba::Agent::removePeer (PPtr< KString > *role*, const k_guid_t & *guid*)

Removes the peer with the given GUID from the given role.

In case the target GUID is not assigned to the given role, this method will complete successfully without making any changes.

Note

A remote agent can be assigned to multiple roles. In that case, it should be removed from each role one at a time.

Parameters

role	The role the peer to be removed from.
guid	The GUID of the peer to be removed.

4.1.3.15 void knorba::Agent::respond (PPtr< Message > *msg*, PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a message in response to a received message.

Note

All messages for which Message::needsResponse() returns `true`, i.e. messages sent using tsendXXX methods, should be responded using this method.

Parameters

msg	The message to reply to.
opcode	The opcode of the response.
content	The content of the response.

4.1.3.16 void knorba::Agent::run ( )

FOR INTERNAL USE.

Never invoke directly.

4.1.3.17 void knorba::Agent::send (const k_guid_t *receiver*, PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a message to another agent.

Parameters

receiver	The GUID of the receiver agent.
opcode	The opcode of the message.
content	The content of the message.

4.1.3.18 void knorba::Agent::send (PPtr< Group > *receivers*, PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a multicast message to a group of agents.

Parameters

receivers	Group of receiver agents.
opcode	The opcode of the message.
content	The content of the message.

4.1.3.19 void knorba::Agent::send (PPtr< KString > *role*, PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a multicast message to all agents with the given role.

Parameters

role	The role of the target agents.
opcode	The opcode of the message.
content	The content of the message.

4.1.3.20 void knorba::Agent::sendToAll (PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a broadcast message.

Parameters

opcode	Message opcode.
content	Message content.

4.1.3.21 void knorba::Agent::sendToLocals (PPtr< KString > *opcode*, PPtr< KValue > *content*)

Sends a broadcast message to all local agents.

Note

Local agents are all the agents running within the same Virtual ARE, plus kernel agents residing on the local machine.

Parameters

opcode	Message opcode.
content	Message content.

4.1.3.22 void knorba::Agent::setPassive (bool *value* = `true`) `[protected]`

Marks this agent as passive.

This method is best to be called once in the constructor. Passive agents will quit automatically when all other non-passive agents quit.

Parameters

value	If set true the agent will be passive, otherwise it will not.

4.1.3.23 void knorba::Agent::sleep (int *msecs*) `[protected]`

Pauses the current thread while making sure the message processor thread is always running.

Parameters

msecs	Amount of time to sleep, measured in milliseconds.

4.1.3.24 Ptr< Message > knorba::Agent::tsend (const k_guid_t *receiver*, PPtr< KString > *opcode*, PPtr< KValue > *content*, k_integer_t *timeout* = -1)

Blocking unicast send.

Sends a message to a remote agent and blocks the current thread until the message is responded or the given timeout expires.

Parameters

receiver	GUID of the receiving agent.
opcode	Message opcode.
content	Message content.
timeout	Expressed in milliseconds. If set to -1 (default value), it will cause this method to wait indefinitely until a response is received. If set to any positive value, this method will wait until a response is received or until the timeout expires, whichever happens sooner.

Returns

The response message, if any, or null pointer if none.

4.1.3.25 Ptr< MessageSet > knorba::Agent::tsend (PPtr< Group > *receivers*, PPtr< KString > *opcode*, PPtr< KValue > *content*, k_integer_t *timeout* = -1)

Blocking multicast send.

Sends a message to a group of remote agents and blocks the current thread until the message is responded by all targets or the given timeout expires.

Parameters

receivers	Group of receiver agents.
opcode	Message opcode.
content	Message content.
timeout	Expressed in milliseconds. If set to -1 (default value), it will cause this method to wait indefinitely until a response is received. If set to any positive value, this method will wait until all targets reply or until the timeout expires, whichever happens sooner.

Returns

A MessageSet containing all responses received.

4.1.3.26 Ptr< MessageSet > knorba::Agent::tsend (PPtr< KString > *receivers*, PPtr< KString > *opcode*, PPtr< KValue > *content*, k_integer_t *timeout* = -1)

Blocking multicast send to peers.

Sends a message to all remote agent with the given role, and blocks the current thread until all target agents respond or the given timeout expires.

Parameters

receivers	The role of receiver peers.
opcode	Message opcode.
content	Message content.
timeout	Expressed in milliseconds. If set to -1 (default value), it will cause this method to wait indefinitely until a response is received. If set to any positive value, this method will wait until a response is received or until the timeout expires, whichever happens sooner.

Returns

A MessageSet containing all responses received.

4.1.3.27 Ptr< MessageSet > knorba::Agent::tsendToLocals (PPtr< KString > *opcode*, PPtr< KValue > *content*, k_integer_t *timeout*)

Blocking local broadcast.

Sends a message to all local agents, and blocks the current thread until the given timeout expires.

Parameters

opcode	Message opcode.
content	Message content.
timeout	Expressed in milliseconds. The amount of time to block the current thread, waiting for responses.

Returns

A MessageSet containing all responses received.

4.1.3.28 void knorba::Agent::unregisterProtocol (Protocol ∗ *p*)

FOR INTERNAL USE.

Do not call directly. Deactivates support for the given protocol.

4.1.4 Member Data Documentation

4.1.4.1 const int knorba::Agent::DEFAULT_QUEUE_SIZE = 16 `[static]`

Default queue size.

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

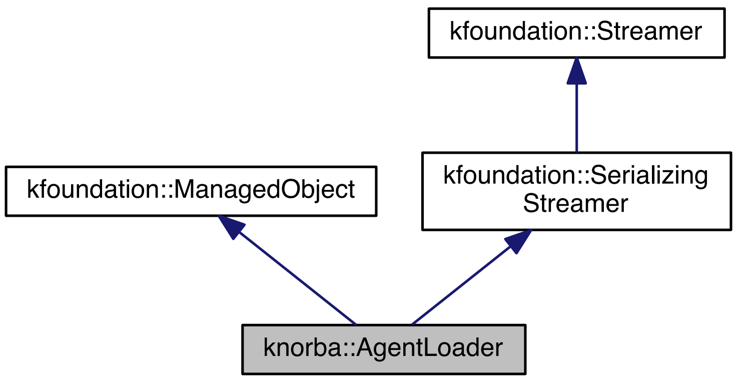
- Agent.h
- Agent.cpp

4.2 knorba::AgentLoader Class Reference

Subclass to make custom agent loaders.

```
#include <knorba/AgentLoader.h>
```

Inheritance diagram for knorba::AgentLoader:



Public Member Functions

- AgentLoader (const string &name, **PPtr**$<$ **Path** $>$ reosources)

  Sole constructor.
- const string & getClassName () const

  Returns the agent class name for this loader.
- **PPtr**$<$ **Path** $>$ getPathToResources () const

  Returns the path to resources directory.

4.2.1 Detailed Description

Subclass to make custom agent loaders.

For applications including embeded systems in which agents are not compiled into individual dynamic libraries, this tool can be used to help ARE to instantiate new agents of a particular type.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 knorba::AgentLoader::AgentLoader (const string & *name*, PPtr$<$ Path $>$ *resources*)

Sole constructor.

Parameters

name	The class name for this agent.
resources	Path to the resouces directory.

4.2.3 Member Function Documentation

4.2.3.1 PPtr$<$ Path $>$ knorba::AgentLoader::getPathToResources ( ) const

Returns the path to resources directory.

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

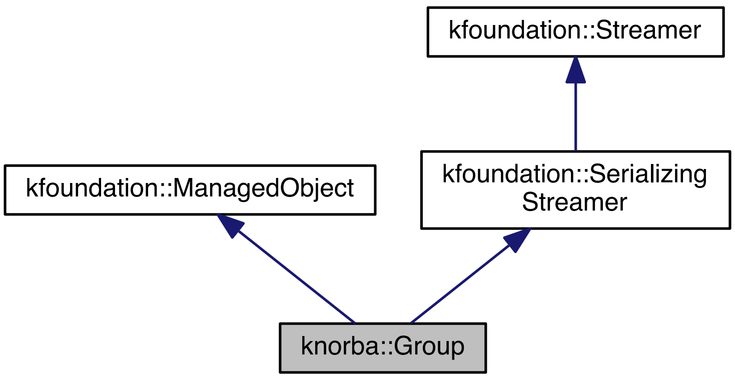
- AgentLoader.h
- AgentLoader.cpp

4.3 knorba::Group Class Reference

Represents a group of agents by their GUIDs.

`#include <knorba/Group.h>`

Inheritance diagram for knorba::Group:



Public Member Functions

- Group ()

  Constructs an empty group.
- void add (const k_guid_t &guid)

  Adds a new GUID, if it is not already added.
- void add (**PPtr**< Group > group)

  Adds all the GUIDs in the given group to this one, not already added.
- void remove (const k_guid_t &guid)

  Removes a GUID from this group, if it exists.
- void clear ()

  Removes all GUIDs in this group.
- int getCount () const

  Returns the number of unique GUIDs in this group.
- const k_guid_t & get (int index) const

  Returns the GUID at the given index.
- bool containts (const k_guid_t &guid) const

  Checks if this group contains the given GUID.
- bool isEmpty () const

  Checks if this group is empty.

Static Public Member Functions

- static **SPtr**< Group > empty_group ()

  Returns a constant empty group.

4.3.1 Detailed Description

Represents a group of agents by their GUIDs.

add() methods prevent GUIDs to be duplicate.

4.3.2 Member Function Documentation

4.3.2.1 void knorba::Group::add (const k_guid_t & *guid*)

Adds a new GUID, if it is not already added.

This method is thread safe.

Parameters

guid	The GUID to add.

4.3.2.2 void knorba::Group::add (PPtr< Group > *group*)

Adds all the GUIDs in the given group to this one, not already added.

This method is thread safe.

Parameters

group	The gruop of GUIDs to add.

4.3.2.3 void knorba::Group::clear ( )

Removes all GUIDs in this group.

This method is thread safe.

4.3.2.4 void knorba::Group::remove (const k_guid_t & *guid*)

Removes a GUID from this group, if it exists.

This method is thread safe.

Parameters

guid	The GUID to remove.

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

- Group.h
- Group.cpp

4.4 knorba::type::k_guid_t Struct Reference

GUID (Gloablly Unique ID)

`#include <definitions.h>`

Public Attributes

- k_appid_t appId

  AppID.
- **kf_int16_t** nodeRank

  Node Rank.
- **kf_int16_t** key

  Public Key.
- k_integer_t lid

  Local ID.

4.4.1 Detailed Description

GUID (Gloablly Unique ID)

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

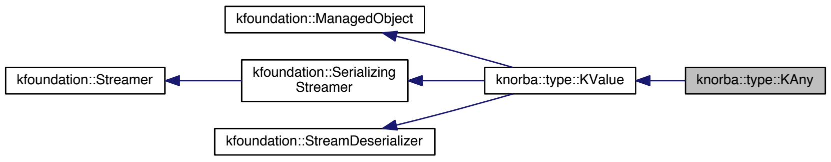
- definitions.h

4.5 knorba::type::KAny Class Reference

Wrapper class and C++ representation of KnoRBA `any` type.

`#include <knorba/type/KAny.h>`

Inheritance diagram for knorba::type::KAny:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KAny
kfoundation::StreamDeserializer

Public Member Functions

- KAny ()

  Constructor; Initiates the stored value with `nothing` *(KValue::NOTHING).*
- KAny (**Ptr**< KValue > value)

  Constructor; Initiates the stored value with the given argument.
- **PPtr**< KValue > getValue () const

  Returns the stored value.
- void setValue (**PPtr**< KValue > v)

  Sets the stored value.
- void setRuntime (Runtime &rt)

  This object needs a reference to runtime in order perform readFromBinaryStream() operation.
- void set (**PPtr**< KValue > other)

Copies the value for this KValue from another one.

- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Writes the internal value by decoding the given InputStream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  This operation is not supported.
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Additional Inherited Members

4.5.1 Detailed Description

Wrapper class and C++ representation of KnoRBA `any` type.

A value of `any` type can store a value of any other type.

4.5.2 Member Function Documentation

4.5.2.1 void knorba::type::KAny::deserialize (**PPtr**< **ObjectToken** > *head*) [virtual]

This operation is not supported.

Implements knorba::type::KValue.

4.5.2.2 void knorba::type::KAny::readFromBinaryStream (**PPtr**< **InputStream** > *input*) [virtual]

Writes the internal value by decoding the given InputStream.

Note

Call setRuntime() before calling this method; otherwise an exception will be thrown.

Parameters

input	The InputStream to read the value from.

Implements knorba::type::KValue.

4.5.2.3 void knorba::type::KAny::set (**PPtr**< **KValue** > *other*) [virtual]

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.5.2.4 void knorba::type::KAny::setRuntime (**Runtime** & *rt*)

This object needs a reference to runtime in order perform readFromBinaryStream() operation.

Parameters

rt	Reference to the current runtime.

4.5.2.5 void knorba::type::KAny::writeToBinaryStream (PPtr< OutputStream > *output*) const [virtual]

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

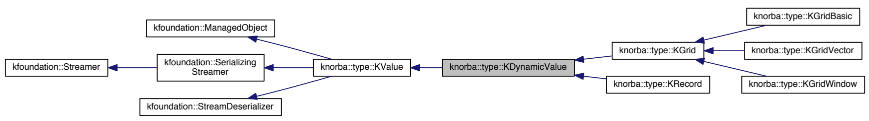
- KAny.h
- KAny.cpp

4.6 knorba::type::KDynamicValue Class Reference

Common denominator of KGrid and KRecord.

`#include <KDynamicValue.h>`

Inheritance diagram for knorba::type::KDynamicValue:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
kfoundation::StreamDeserializer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KGrid
knorba::type::KRecord
knorba::type::KGridBasic
knorba::type::KGridVector
knorba::type::KGridWindow

Additional Inherited Members

4.6.1 Detailed Description

Common denominator of KGrid and KRecord.

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

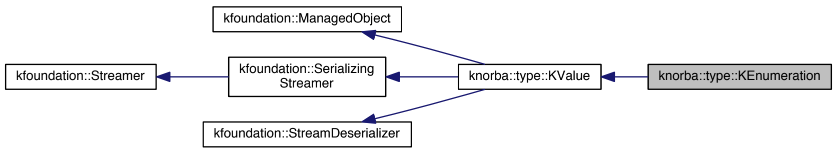
- KDynamicValue.h

4.7 knorba::type::KEnumeration Class Reference

Wrapper class and C++ representation for KnoRBA `enumeration`.

`#include <knorba/type/KEnumeration.h>`

Inheritance diagram for knorba::type::KEnumeration:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KEnumeration
kfoundation::StreamDeserializer

Public Member Functions

- KEnumeration (**PPtr**< KEnumerationType > type)

  Constructor; initiates the stored value with the first enumeration member.
- KEnumeration (**PPtr**< KEnumerationType > type, const k_octet_t ordinal)

  Constructor; initializes the stored value with the given ordinal.
- KEnumeration (**PPtr**< KEnumerationType > type, const string &label)

  Constructor; initializes the stored value with the given label.
- virtual k_octet_t getOrdinal () const

  Returns the ordinal of the stored value.
- string getLabel () const

  Returns the label for the stored value.
- virtual void set (const k_octet_t value)

  Sets the stored value with the given ordinal.
- void set (const string &value)

  Sets the stored value with the given label.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Additional Inherited Members

4.7.1 Detailed Description

Wrapper class and C++ representation for KnoRBA `enumeration`.

To use, the corresponding KEnumerationType shold be defined in advance:

```
Ptr<KEnumerationType> fruitEnum = new KEnumerationType("Fruit");
fruitEnum->addMember("apple")
         ->addMember("orange")
         ->addMember("banana");

Ptr<KEnumeration> myFruit = new KEnumeration(fruitEnum, "apple");
```

4.7.2 Constructor & Destructor Documentation

4.7.2.1 knorba::type::KEnumeration::KEnumeration (PPtr< KEnumerationType > *type*)

Constructor; initiates the stored value with the first enumeration member.

Parameters

type	The type for the stored value.

4.7.2.2 knorba::type::KEnumeration::KEnumeration (PPtr< KEnumerationType > *type*, const k_octet_t *ordinal*)

Constructor; initializes the stored value with the given ordinal.

Parameters

type	The type for the stored value.
ordinal	Ordinal of the initial value.

4.7.2.3 knorba::type::KEnumeration::KEnumeration (PPtr< KEnumerationType > *type*, const string & *label*)

Constructor; initializes the stored value with the given label.

Parameters

type	The type for the stored value.
value	Label of the initial value.

4.7.3 Member Function Documentation

4.7.3.1 void knorba::type::KEnumeration::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.7.3.2 void knorba::type::KEnumeration::set (PPtr< KValue > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.7.3.3 void knorba::type::KEnumeration::writeToBinaryStream (PPtr< OutputStream > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

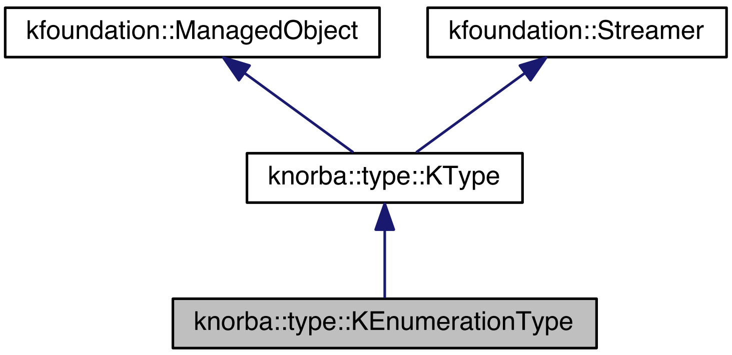
- KEnumeration.h
- KEnumeration.cpp

4.8 knorba::type::KEnumerationType Class Reference

Instantiate to create custom KnoRBA **enumeration** type.

```
#include <knorba/type/KEnumerationType.h>
```

Inheritance diagram for knorba::type::KEnumerationType:



Public Member Functions

- KEnumerationType (const string &name)

  Constructor.
- **PPtr**< KEnumerationType > addMember (k_octet_t ordinal, const string &label)

  Adds a member to this enumeration, associating it to an ordinal.
- **PPtr**< KEnumerationType > addMember (const string &label)

  Adds a member to this enumeration, automatically assigning an ordinal to the given label.
- string getLabelForOrdinal (const k_octet_t ordinal) const

  Returns the label associated with the given ordinal.
- int getOrdinalForLabel (const string &label) const

  Returns the ordinal associated with the given label.
- k_octet_t getNumberOfMembers () const

  Returns the number of members of this enumeration.
- k_octet_t getMaxOrdinal () const

  Returns the maximum ordinal in this enumeration.
- **Array**< k_octet_t >::Ptr_t getAllOrdinals () const

Returns an array containing ordinals of all members of this enumeration.

- string getLabelForMemberAtIndex (const k_octet_t index) const

  Returns the label for the memebr at the given index.
- k_octet_t getOrdinalForMemberAtIndex (const k_octet_t index) const

  Returns the ordinal for the member at the given index.
- k_octet_t getOrdinalForValueAtAddress (const k_octet_t ∗const addr) const

  Returns the ordinal for the enumeration value stored at the given memory location.
- string getLabelForValueAtAddress (const k_octet_t ∗const addr) const

  Returns the label for the enumeration value stored at the given memory location.
- void setValueAtAddressWithOrdinal (k_octet_t ∗const addr, const k_octet_t ordinal) const

  Stores an enumeration value with the given ordinal at the given memory location.
- void setValueAtAddressWithLabel (k_octet_t ∗const addr, const string &label) const

  Stores an enumeration value with the given label at the given memory address.
- bool isCastableTo (**PPtr**< KType > t) const

  Checks if the type represented by this object is castable to the given type.
- bool isAutomaticCastableTo (**PPtr**< KType > t) const

  Checks if this type can be automatically casted to the given type by KnoIL language interpreter.
- bool equals (**PPtr**< KType > t) const

  Checks if type represented by this object is equivalant to the one represented by the given argument.
- int getSizeInOctets () const

  If hasConstantSize() returns `true`, this method returns the amount of octets a value of this type consumes when stored in memory or sent over a stream; otherwise it resturns 0.
- bool isPrimitive () const

  Returns `true` iif this object represents a primitive type.
- bool hasConstantSize () const

  Returns true iif the type represented by this object has constant size.
- **Ptr**< KValue > instantiate () const

  Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object.
- string toKnois () const

  Returns type description in KnoIS language.

Additional Inherited Members

4.8.1 Detailed Description

Instantiate to create custom KnoRBA **enumeration** type.

Usage:

```
Ptr<KEnumerationType> myEnum = new KEnumerationType("season");
myEnum->addMember("spring")
      ->addMember("summer")
      ->addMember("fall")
      ->addMember("winter");
```

To instantiate,

```
Ptr<KEnumeration> value = new KEnumeration(myEnum);
```

or

```
Ptr<KEnumeration> value = myEnum->instantiate().AS(KEnumeration);
```

4.8.2 Constructor & Destructor Documentation

4.8.2.1 knorba::type::KEnumerationType::KEnumerationType (const string & *name*)

Constructor.

Parameters

name	Name for the custom enumeration type.

4.8.3 Member Function Documentation

4.8.3.1 PPtr< KEnumerationType > knorba::type::KEnumerationType::addMember (k_octet_t *ordinal,* const string & *label*)

Adds a member to this enumeration, associating it to an ordinal.

Parameters

ordinal	Ordinal for the new member.
label	Label for the new member.

4.8.3.2 PPtr< KEnumerationType > knorba::type::KEnumerationType::addMember (const string & *label*)

Adds a member to this enumeration, automatically assigning an ordinal to the given label.

The chosen ordinal number equals maximum ordinal plus one.

Parameters

label	Label for the new member.

4.8.3.3 bool knorba::type::KEnumerationType::equals (PPtr< KType > *t*) const [virtual]

Checks if type represented by this object is equivalant to the one represented by the given argument.

Checks if this object and the given argument represent the same type.

Reimplemented from knorba::type::KType.

4.8.3.4 string knorba::type::KEnumerationType::getLabelForMemberAtIndex (const k_octet_t *index*) const

Returns the label for the memebr at the given index.

Parameters

index	Index, a value between 0 and getNumberOfMembers() - 1.

4.8.3.5 string knorba::type::KEnumerationType::getLabelForOrdinal (const k_octet_t *ordinal*) const

Returns the label associated with the given ordinal.

Returns an empty string if given an invalid ordinal.

Parameters

ordinal	The ordinal to find label for.

4.8.3.6 string knorba::type::KEnumerationType::getLabelForValueAtAddress (const k_octet_t ∗const *addr*) const

Returns the label for the enumeration value stored at the given memory location.

Parameters

addr	Pointer to a memory location storing an enumeration value.

4.8.3.7 int knorba::type::KEnumerationType::getOrdinalForLabel (const string & *label*) const

Returns the ordinal associated with the given label.

Returns -1 if there is no such label.

Parameters

label	The label to find the ordinal for.

4.8.3.8 k_octet_t knorba::type::KEnumerationType::getOrdinalForMemberAtIndex (const k_octet_t *index*) const

Returns the ordinal for the member at the given index.

Parameters

index	Index, a value between 0 and getNumberOfMembers() - 1.

4.8.3.9 k_octet_t knorba::type::KEnumerationType::getOrdinalForValueAtAddress (const k_octet_t ∗const *addr*) const

Returns the ordinal for the enumeration value stored at the given memory location.

Parameters

addr	Pointer to a memory location storing an enumeration value.

4.8.3.10 bool knorba::type::KEnumerationType::hasConstantSize ( ) const [virtual]

Returns true iif the type represented by this object has constant size.

Types with variable size are `string` (KType::STRING), `raw` (KType::RAW), and `grid` (KGridType). The rest of them have constant sizes.

Implements knorba::type::KType.

4.8.3.11 Ptr< KValue > knorba::type::KEnumerationType::instantiate ( ) const [virtual]

Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object. For example,

```
KType::INTEGER->instantiate()
```

will return an instance of KInteger.

Implements knorba::type::KType.

4.8.3.12 void knorba::type::KEnumerationType::setValueAtAddressWithLabel (k_octet_t ∗const *addr*, const string & *label*) const

Stores an enumeration value with the given label at the given memory address.

Parameters

addr	Pointer to a preallocated memory location.
label	The label of the value to store.

4.8.3.13 void knorba::type::KEnumerationType::setValueAtAddressWithOrdinal (k_octet_t ∗const *addr*, const k_octet_t *ordinal*) const

Stores an enumeration value with the given ordinal at the given memory location.

Parameters

addr	Pointer to a preallocated memory location.
ordinal	The ordinal of the value to store.

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

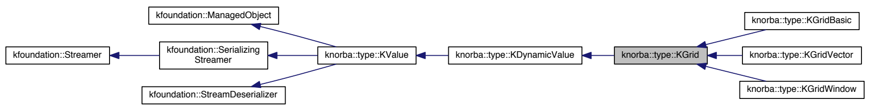
- KEnumerationType.h
- KEnumerationType.cpp

4.9 knorba::type::KGrid Class Reference

Wrapper class and C++ representation of KnoRBA `grid` type.

`#include <knorba/type/KGrid.h>`

Inheritance diagram for knorba::type::KGrid:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
kfoundation::StreamDeserializer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KGrid
knorba::type::KGridBasic
knorba::type::KGridVector
knorba::type::KGridWindow

Public Member Functions

- **PPtr**< KRecord > at (const **Tuple** &index, **PPtr**< KRecord > wrapper) const throw (IndexOutOfBoundException)

  Accessor method.
- KRecord & at (const **Tuple** &index, KRecord &wrapper) const throw (IndexOutOfBoundException)

  High-speed equivalant for (const Tuple&, PPtr<KRecord>).
- void copyFrom (const **PPtr**< KGrid > src, const **Tuple** &srcOffset, const **Tuple** &dstOffset, const **Tuple** &size)

  Copies the values of the given range of cells from the given offset of another grid to the given offset of this grid.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.

- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Additional Inherited Members

4.9.1 Detailed Description

Wrapper class and C++ representation of KnoRBA `grid` type.

KnoRBA `grid` is a multi-dimensional array of records. Both KGrid and KRecord are optimized together for high performance computing. KGrid takes full advantage of range arithmatics objects in KFoundation.

All values of a grid are squizzed into a continues portion of memory for better allocation and cache performance. No KRecord object is allocated internally. Allocating a KRecord for each cell would consume too much uncessary memory, and also cause fragmentation, decreasing cache performance. Therefore the design pattern chosen for this object is *sliding record*, that is, only one KRecord is allocated and is slid over desired cells of the grid. However the KRecord is not allocated internally because it would make it impossible to read and write on the same grid using more than one thread. Therefore, eahc thread should allocate the slidding KRecord externally.

KGrid itslef is an abstract class. Use one of the implementations provided in the same header:

- KGridBasic – internally allocated multi-dimensional array of cells.
- KGridWindow – a sub array of an externally allocated KGrid.
- KGridVector – a one dimensional dynamically resizable array of cells.

The basic usage of KGrid with simple record type is as follows:

```
#include <kfoundation/Tuple.h>
#include <kfoundation/RangeIterator.h>
using namespace kfoundation;

Ptr<KGridType> gridType = new KGridType(KType::REAL, 2);
Ptr<KGrid> grid = new KGridBasic(gridType, Tuple(100, 100));
Ptr<KRecord> r = new KRecord(grid);
for(RangeIterator i(grid->getSize()); i.hasMore(); i.next()) {
  grid->at(i, r)->setInteger(i.at(0));
}
```

The at() method slides the given KRecord on to the cell at the given index.

```
grid->at(Tuple2D(11, 12), r);
r->setInteger(124);
```

In case the cell record type has more than one field, the name or index of the field can be supplied to KRecord::set-XXX() and KRecord::getXX() methods.

```
r->setInteger("year", 1981);
```

4.9.2 Member Function Documentation

4.9.2.1 PPtr< KRecord > knorba::type::KGrid::at (const Tuple & *index*, PPtr< KRecord > *wrapper*) const throw IndexOutOfBoundException)

Accessor method.

Slides the given wrapper record onto the cell at the given index. The given KRecord should be created using KRecord::KRecord(PPtr<KGrid>) with this object given as argument.

Note

For performance reasons valid index range is not hardly enforced.

4.9.2.2 KRecord & knorba::type::KGrid::at (const Tuple & *index*, KRecord & *wrapper*) const throw IndexOutOfBoundException)

High-speed equivalant for (const Tuple&, PPtr<KRecord>).

Note

For performance reasons valid index range is not hardly enforced.

4.9.2.3 void knorba::type::KGrid::copyFrom (const PPtr< KGrid > *src*, const Tuple & *srcOffset*, const Tuple & *dstOffset*, const Tuple & *size*)

Copies the values of the given range of cells from the given offset of another grid to the given offset of this grid.

Parameters

src	The grid to copy values from.
srcOffset	Source offset.
dstOffset	Destination offset.
size	Size of the range of values to copy.

4.9.2.4 void knorba::type::KGrid::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.9.2.5 void knorba::type::KGrid::set (PPtr< KValue > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.9.2.6 void knorba::type::KGrid::writeToBinaryStream (PPtr< OutputStream > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

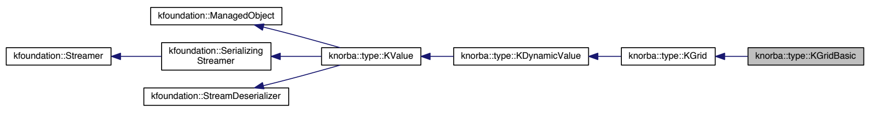
- KGrid.h
- KGrid.cpp

4.10 knorba::type::KGridBasic Class Reference

Basic variant of KGrid.

`#include <knorba/type/KGrid.h>`

Inheritance diagram for knorba::type::KGridBasic:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
kfoundation::StreamDeserializer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KGrid
knorba::type::KGridBasic

Public Member Functions

- KGridBasic (**PPtr**< KGridType > type)

  Constructor; creates a 0-dimensional grid with 0 cells.
- KGridBasic (**PPtr**< KGridType > type, const **Tuple** &dims, bool clear=false)

  Constructor; internally allocates a grid of the given dimensions.
- ∼KGridBasic ()

  Deconstructor.

Additional Inherited Members

4.10.1 Detailed Description

Basic variant of KGrid.

Most often, this is the class to use for creating and manipulating KnoRBA `grid`.

Read documentation for KGrid for more details.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 knorba::type::KGridBasic::KGridBasic (PPtr< KGridType > *type*, const Tuple & *dims*, bool *clear* = `false`)

Constructor; internally allocates a grid of the given dimensions.

Parameters

type	Grid type.
dims	Grid dimensions.
clear	Optional. If set `true` initiates the cells with zeros. Setting this parameter to `false` will save some execution time. Default value is `false`.

4.10.2.2 knorba::type::KGridBasic::∼KGridBasic ( )

Deconstructor.

Frees internally allocated memory.

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

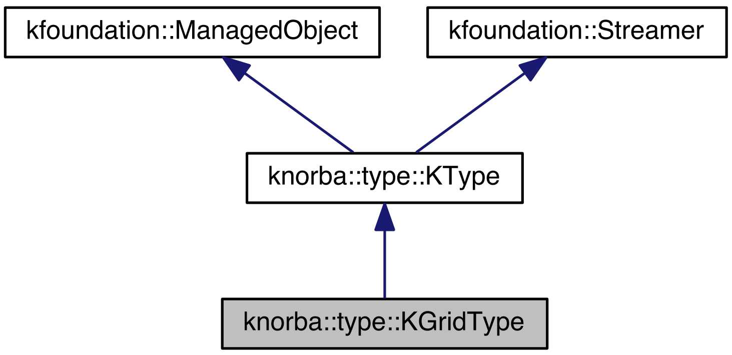
- KGrid.h
- KGrid.cpp

4.11 knorba::type::KGridType Class Reference

Instantiate to create a custom KnoRBA **grid** type.

`#include <knorba/type/KGridType.h>`

Inheritance diagram for knorba::type::KGridType:



Public Member Functions

- KGridType (**PPtr**< KType > recordTypes, k_octet_t nDimensions)

  Constructs a representation of a custom KnoRBA grid with the given cell type and given number of dimensions.
- KGridType (**PPtr**< KRecordType > recordTypes, k_octet_t nDimensions)

  Constructs a representation of a custom KnoRBA grid type with simple cell type.
- **PPtr**< KRecordType > getRecordType () const

  Returns the type of grid cells.
- short int getNDimensions () const

  Returns the number of dimensions.
- bool isCastableTo (**PPtr**< KType > t) const

  Checks if the type represented by this object is castable to the given type.
- bool isAutomaticCastableTo (**PPtr**< KType > t) const

  Checks if this type can be automatically casted to the given type by KnoIL language interpreter.
- bool equals (**PPtr**< KType > t) const

  Checks if type represented by this object is equivalant to the type represented by the given argument.
- int getSizeInOctets () const

  If hasConstantSize() returns `true`, this method returns the amount of octets a value of this type consumes when stored in memory or sent over a stream; otherwise it resturns 0.
- bool isPrimitive () const

  Returns `true` iif this object represents a primitive type.
- bool hasConstantSize () const

  Returns true iif the type represented by this object has constant size.
- **Ptr**< KValue > instantiate () const

  Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object.
- string toKnois () const

  Returns type description in KnoIS language.

Additional Inherited Members

4.11.1 Detailed Description

Instantiate to create a custom KnoRBA **grid** type.

A KnoRBA grid is a multi-dimensional array of KnoRBA **record**. Therefore, creating a grid type often involves creating a record type in advance:

```
Ptr<KRecordType> myRecordType = new KRecordType("MyRecordType");
myRecordType->addField("pressure", KType::REAL)
            ->addField("temperature", KType::REAL);
Ptr<KGridType> myGridType = new KGridType(myRecordType, 3);
```

In the special case that each cell of the grid has a simple value, there is a shortcut constructor:

```
Ptr<KGridType> myGridTupe = new KGridType(KType::INTEGER, 2);
```

In the above case, a new instance of KRecordType will be created internally which can be retrieved via getRecordType() method.

Note

In KnoRBA strongly-typed system, two grid types with the same record type but different number of dimensions are considered as different types.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 knorba::type::KGridType::KGridType (PPtr< KType > *recordType,* k_octet_t *nDimension*)

Constructs a representation of a custom KnoRBA grid with the given cell type and given number of dimensions.

Parameters

recordType	The type of grid cells.
nDimensions	Number of grid dimensions.

4.11.2.2 knorba::type::KGridType::KGridType (PPtr< KRecordType > *recordType,* k_octet_t *nDimension*)

Constructs a representation of a custom KnoRBA grid type with simple cell type.

This method will construct an instance of KRecordType internally, with only one field who's type is the given argument.

Parameters

recordType	The type of the sole record member of each cell.
nDimensions	Number of grid dimensions.

4.11.3 Member Function Documentation

4.11.3.1 bool knorba::type::KGridType::equals (PPtr< KType > *t*) const [virtual]

Checks if type represented by this object is equivalant to the type represented by the given argument.

Returns `true` iff (1) `t` is an instance of KGridType, (2) record type of the grid type represented by this object is equivalant of the record type of the given grid type, and (3) the number of dimensions of this object equals to that of the given object.

Reimplemented from knorba::type::KType.

4.11.3.2 bool knorba::type::KGridType::hasConstantSize ( ) const [virtual]

Returns true iif the type represented by this object has constant size.

Types with variable size are `string` (KType::STRING), `raw` (KType::RAW), and `grid` (KGridType). The rest of them have constant sizes.

Implements knorba::type::KType.

4.11.3.3 Ptr< KValue > knorba::type::KGridType::instantiate ( ) const [virtual]

Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object. For example,

```
KType::INTEGER->instantiate()
```

will return an instance of KInteger.

Implements knorba::type::KType.

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

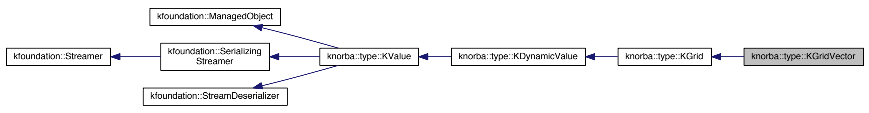
- KGridType.h
- KGridType.cpp

4.12 knorba::type::KGridVector Class Reference

One-dimensional variable-length flavour of KGrid.

`#include <knorba/type/KGrid.h>`

Inheritance diagram for knorba::type::KGridVector:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KGrid
knorba::type::KGridVector
kfoundation::StreamDeserializer

Public Member Functions

- **PPtr**< KRecord > add (**PPtr**< KRecord > wrapper)

    Adds a new record to the top of this vector, and slides the given wrapper record on it.
- **PPtr**< KRecord > insert (**PPtr**< KRecord > wrapper, const k_integer_t index)

    Adds a new record at the given index and slides the given wrapper record on it.
- void remove (const k_integer_t index)

    Removes the record at the given index, shifting all records at higher indexes downwards.
- void removeLast ()

    Removes the last record.
- **PPtr**< KRecord > last (**PPtr**< KRecord > wrapper) const

    Slides the given wrapper record on the last item in the vector.
- void clear ()

    Removes all elements in this record.
- k_integer_t getNElements () const

    Returns the number of elements in the vector.

Additional Inherited Members

4.12.1 Detailed Description

One-dimensional variable-length flavour of KGrid.

This `grid` implementaion has only one dimension but in return, it has dynamic size and supports common vector operations.

Read documentation for KGrid for more details.

4.12.2 Member Function Documentation

4.12.2.1 PPtr< KRecord > knorba::type::KGridVector::add (PPtr< KRecord > *wrapper*)

Adds a new record to the top of this vector, and slides the given wrapper record on it.
Usage:

```
vector->add(record)->setInteger(128);
```

Returns

Same pointer as given arugment.

4.12.2.2 PPtr< KRecord > knorba::type::KGridVector::insert (PPtr< KRecord > *wrapper,* const k_integer_t *index*)

Adds a new record at the given index and slides the given wrapper record on it.

Usage:

vector->insert(record, 4)->setInteger(128);

Returns

Same pointer as the first argument.

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

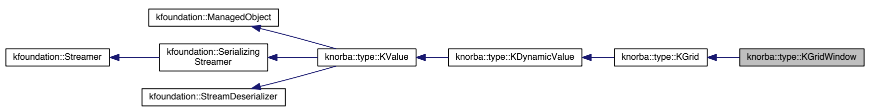
- KGrid.h
- KGrid.cpp

4.13 knorba::type::KGridWindow Class Reference

Places a virtual index range over a portion of an existing grid.

`#include <knorba/type/KGrid.h>`

Inheritance diagram for knorba::type::KGridWindow:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
kfoundation::StreamDeserializer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KGrid
knorba::type::KGridWindow

Public Member Functions

- void setSource (**PPtr**< KGrid > physical)

  Changes the source physical grid to the given one.
- void setWindow (const **Range** &physicalRange)

  Changes the virtual size of this grid window.
- void setWindow (const **Range** &physicalRange, const **Tuple** &virtualOffset)

  Changes the source and virtual range of this window.
- **Tuple** getVirtualOffset () const

  Returns this window's virtual offset.
- **Range** getVirtualRagne () const

  Returns this window's virtual range.
- **PPtr**< KRecord > atVirtual (const **Tuple** &index, **PPtr**< KRecord > wrapper) const

  Slides the given wrapper record on to the given virtual index.
- KRecord & atVirtual (const **Tuple** &index, KRecord &wrapper) const

  High-speed alternative for atVirtual(const Tuple&, PPtr<KRecord>).

Additional Inherited Members

4.13.1 Detailed Description

Places a virtual index range over a portion of an existing grid.

This class is specially usefull in high-performance scenarios when exchanging boundaries between nodes, and computing over a range of values using multiple nodes and multiple threads.

Read documentation for KGrid for more details.

4.13.2 Member Function Documentation

4.13.2.1 PPtr< KRecord > knorba::type::KGridWindow::atVirtual (const Tuple & *index*, PPtr< KRecord > *wrapper*) const

Slides the given wrapper record on to the given virtual index.

The given record should be created using KRecord::KRecord(PPtr<KGrid>) given this object as its argument.

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

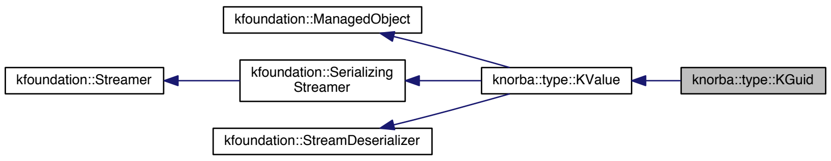
- KGrid.h
- KGrid.cpp

4.14 knorba::type::KGuid Class Reference

Wrapper class for KnoRBA `GUID` (Globally Unique Identifier).

`#include <knorba/type/KGuid.h>`

Inheritance diagram for knorba::type::KGuid:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KGuid
kfoundation::StreamDeserializer

Public Member Functions

- KGuid ()

  Constructor; initiates the stored value with zero().
- KGuid (const k_guid_t &v)

  Constructor; initiates the stored value with the given argument.
- virtual k_guid_t get () const

  Returns the stored value.
- virtual void set (const k_guid_t &v)

  Sets the stored value to the given value.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Member Functions

- static const k_guid_t & zero ()

  Returns reference to internally stored zero constant.
- static void randomizeAppId (k_guid_t &target)

  Rewrites the AppID part of the given argument with a random value.
- static void randomizeKey (k_guid_t &target)

  Replaces the key part of the given argument with a random number.
- static bool areOnTheSameApp (const k_guid_t &first, const k_guid_t &second)

  Returns `true` *iff the two given arguments have the same AppID.*
- static bool areOnTheSameNode (const k_guid_t &first, const k_guid_t &second)

  Returns `true` *iff two given arguments have the same AppID and node rank.*
- static string toString (const k_guid_t &value)

  Returns the string representation of the given value.

- static string toShortString (const k_guid_t &value)

  Returns a string containing NodeRank and LocalId parts of the given value.
- static string appIdToString (const k_guid_t &value)

  Returns the AppId part of the given value as a string.

Additional Inherited Members

4.14.1 Detailed Description

Wrapper class for KnoRBA `GUID` (Globally Unique Identifier).

GUID is a segmented 8-bit value as represented by `knorba::type::k_guid_t`.

4.14.2 Member Function Documentation

4.14.2.1 void knorba::type::KGuid::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.14.2.2 void knorba::type::KGuid::set (PPtr< KValue > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.14.2.3 void knorba::type::KGuid::writeToBinaryStream (PPtr< OutputStream > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

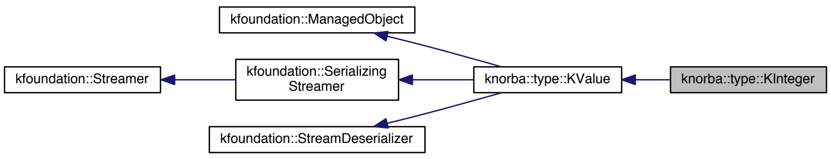
- KGuid.h
- KGuid.cpp

4.15 knorba::type::KInteger Class Reference

Wrapper class form KnoRBA `integer` type.

```
#include <knorba/type/KInteger.h>
```

Inheritance diagram for knorba::type::KInteger:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KInteger
kfoundation::StreamDeserializer

Public Member Functions

- KInteger ()

  Constructor; sets the stored value to 0.
- KInteger (const k_integer_t v)

  Constructor; sets the stored value to the given argument.
- virtual k_integer_t get () const

  Returns the stored value.
- virtual void set (const k_integer_t v)

  Sets the stored value.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Attributes

- static const k_integer_t MAX_VALUE = 2147483647

  The maximum possible value of KnoRBA `integer`.
- static const k_integer_t MIN_VALUE = -2147483647

  The minimum possible value of KnoRBA `integer`

4.15.1 Detailed Description

Wrapper class form KnoRBA `integer` type.

A value of type `integer` is a 32-bit (4-octet) 2's complement signed integer between KInteger::MAX_VALUE and KInteger::MIN_VALUE. The scalar type associated with this class is `knorba::type::k_integer_t`.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 knorba::type::KInteger::KInteger (const k_integer_t *v*)

Constructor; sets the stored value to the given argument.

Parameters

v	The initial value.

4.15.3 Member Function Documentation

4.15.3.1 void knorba::type::KInteger::readFromBinaryStream (PPtr< InputStream > *input*) [virtual]

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.15.3.2 void knorba::type::KInteger::set (const k_integer_t *v*) [virtual]

Sets the stored value.

Parameters

v	The value to set to.

4.15.3.3 void knorba::type::KInteger::set (PPtr< KValue > *other*) [virtual]

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.15.3.4 void knorba::type::KInteger::writeToBinaryStream (PPtr< OutputStream > *output*) const [virtual]

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

4.15.4 Member Data Documentation

4.15.4.1 const k_integer_t knorba::type::KInteger::MAX_VALUE = 2147483647 [static]

The maximum possible value of KnoRBA `integer`.

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

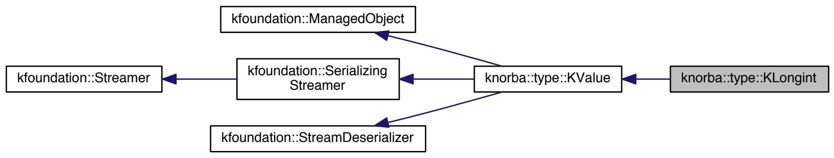
- KInteger.h
- KInteger.cpp

4.16 knorba::type::KLongint Class Reference

Wrapper class for KnoRBA `longint` type.

```
#include <knorba/type/KLongint.h>
```

Inheritance diagram for knorba::type::KLongint:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KLongint
kfoundation::StreamDeserializer

Public Member Functions

- KLongint ()

  Constructor; sets the stored value to 0.
- KLongint (const k_longint_t v)

  Constructor; sets the stored value to the given argument.
- virtual void set (const k_longint_t v)

  Sets the stored value.
- virtual k_longint_t get () const

  Returns the stored value.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Attributes

- static const k_longint_t MIN_VALUE = -9223372036854775807

  The minimum possible value for `longint` type.
- static const k_longint_t MAX_VALUE = 9223372036854775807

  The maximum possible value for `longint` type.

4.16.1 Detailed Description

Wrapper class for KnoRBA `longint` type.

A value of `longint` type is a 2's complement 64-bit (8-octet) signed integer between KLongint::MIN_VALUE and KLongint::MAX_VALUE. The scalar type associated with this class is `knorba::type::k_longint_t`.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 knorba::type::KLongint::KLongint (const k_longint_t *v*)

Constructor; sets the stored value to the given argument.

Parameters

v	The value to set to.

4.16.3 Member Function Documentation

4.16.3.1 void knorba::type::KLongint::readFromBinaryStream (PPtr< InputStream > *input*) [virtual]

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.16.3.2 void knorba::type::KLongint::set (const k_longint_t *v*) [virtual]

Sets the stored value.

Parameters

v	The value to set to.

4.16.3.3 void knorba::type::KLongint::set (PPtr< KValue > *other*) [virtual]

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.16.3.4 void knorba::type::KLongint::writeToBinaryStream (PPtr< OutputStream > *output*) const [virtual]

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

4.16.4 Member Data Documentation

4.16.4.1 const k_longint_t knorba::type::KLongint::MAX_VALUE = 9223372036854775807 [static]

The maximum possible value for `longint` type.

4.16.4.2 const k_longint_t knorba::type::KLongint::MIN_VALUE = -9223372036854775807 [static]

The minimum possible value for `longint` type.

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

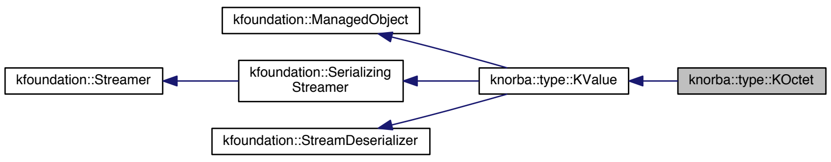
- KLongint.h
- KLongint.cpp

4.17 knorba::type::KOctet Class Reference

Wrapper class for KnoRBA `octet` type.

`#include <knorba/type/KOctet.h>`

Inheritance diagram for knorba::type::KOctet:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KOctet
kfoundation::StreamDeserializer

Public Member Functions

- KOctet ()

  Constructor; sets the stored value to 0.
- KOctet (const k_octet_t v)

  Constructor; sets the stored value to the given argument.
- virtual k_octet_t get () const

  Returns the stored value.
- virtual void set (const k_octet_t v)

  Sets the stored value.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Member Functions

- static k_octet_t parseHex (char ch)

  Parses a k_octet_t value from a single digit hexadecimal representation.
- static k_octet_t parseHex (const char ∗chars)

  Parses a k_octet_t from a hexadecimal representation stored in a c-style string.

Additional Inherited Members

4.17.1 Detailed Description

Wrapper class for KnoRBA `octet` type.

A value of type `octet` is an 8-bit unsigned integer between 0 and 255. Scalar type associated with this class is `knorba::type::k_octet_t`.

Implementation transparency is KnoRBA is partially achieved by having every value to be expressed by a sequence of octets.

4.17.2 Member Function Documentation

4.17.2.1 void knorba::type::KOctet::readFromBinaryStream (**PPtr< InputStream >** *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.17.2.2 void knorba::type::KOctet::set (const k_octet_t *v*) `[virtual]`

Sets the stored value.

Parameters

v	The value to set to.

4.17.2.3 void knorba::type::KOctet::set (**PPtr< KValue >** *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.17.2.4 void knorba::type::KOctet::writeToBinaryStream (**PPtr< OutputStream >** *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

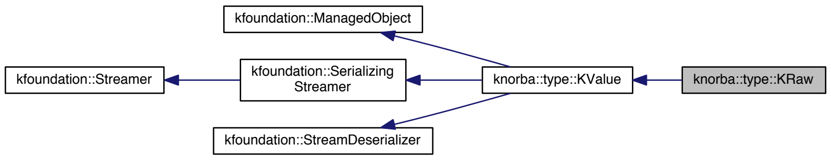
- KOctet.h
- KOctet.cpp

4.18 knorba::type::KRaw Class Reference

Wrapper class and C++ representation for KnoRBA `raw` type.

```
#include <knorba/type/KRaw.h>
```

Inheritance diagram for knorba::type::KRaw:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KRaw
kfoundation::StreamDeserializer

Public Member Functions

- KRaw ()

  Constructor; initializes the stored value with a raw string of size 0.
- ∼KRaw ()

  Deconstructor.
- void set (const k_octet_t ∗data, const k_longint_t size)

  Sets the internally stored value to the given buffer.
- const k_octet_t ∗ getData () const

  Returns pointer to the begining of the internal buffer.
- k_longint_t getNOctets () const

  Returns the number of octets of the stored data.
- void readDataFromFile (**PPtr**< **Path** > path)

  Sets the data stored in this object from the contents of the given file.
- void writeDataToFile (**PPtr**< **Path** > path)

  Writes the stored data into the file at the given path.
- **Ptr**< **BufferInputStream** > getDataAsInputStream () const

  Returns an InputStream that contains the data stored in this object.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  KRaw does not support this operation.
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Additional Inherited Members

4.18.1 Detailed Description

Wrapper class and C++ representation for KnoRBA `raw` type.

A value of raw type is a continues sequence of arbitrary octets.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 knorba::type::KRaw::∼KRaw ( )

Deconstructor.

Deletes the internally allocated buffer.

4.18.3 Member Function Documentation

4.18.3.1 void knorba::type::KRaw::deserialize (PPtr< ObjectToken > *headToken*) `[virtual]`

KRaw does not support this operation.

Implements knorba::type::KValue.

4.18.3.2 void knorba::type::KRaw::readDataFromFile (PPtr< Path > *path*)

Sets the data stored in this object from the contents of the given file.

Parameters

path	Path to the file to read.

4.18.3.3 void knorba::type::KRaw::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.18.3.4 void knorba::type::KRaw::set (const k_octet_t ∗ *data*, const k_longint_t *size*)

Sets the internally stored value to the given buffer.

Parameters

data	The buffer to copy data from.
size	The number of octets to copy.

4.18.3.5 void knorba::type::KRaw::set (PPtr< KValue > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.18.3.6 void knorba::type::KRaw::writeDataToFile (PPtr< Path > *path*)

Writes the stored data into the file at the given path.

Parameters

path	Path to the file to write to.

4.18.3.7 void knorba::type::KRaw::writeToBinaryStream (**PPtr**< **OutputStream** > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

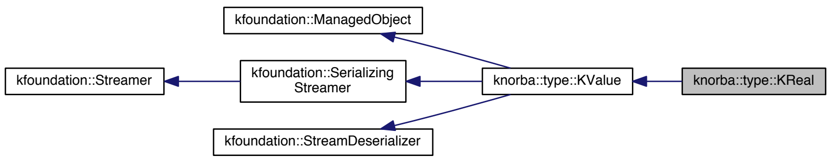
- KRaw.h
- KRaw.cpp

4.19 knorba::type::KReal Class Reference

Wrapper class for KnoRBA `real` type.

`#include <knorba/type/KReal.h>`

Inheritance diagram for knorba::type::KReal:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KReal
kfoundation::StreamDeserializer

Public Member Functions

- KReal ()

  Constructor; initiates the stored value with 0.
- KReal (const k_real_t v)

  Constructor; initiates the sotred value with the given argument.
- virtual void set (const k_real_t v)

  Sets the stored value to the one provided.
- virtual k_real_t get () const

  Returns the stored value.
- bool isNaN () const

  Returns `true` iff the stored value is NaN.
- bool isInfinity () const

  Returns `true` iff the stored value is positive or negative infinity.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.

- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Attributes

- static const k_real_t INFINITY = 0x7FF0000000000000

  IEEE 754 representation of positive infinity.
- static const k_real_t NAN = 0x7FFFFFFFFFFFFFFF

  IEEE 754 representation of not-a-number (NaN).

4.19.1 Detailed Description

Wrapper class for KnoRBA `real` type.

A value of `real` type is a 64-bit (8-octet) IEEE 754 floating point number. The scalar type associated with this class is `knorba::type::k_real_t`. Special values, NaN (not a number), and infinity, are stored in `KReal::NAN` and `KReal::INFINITY` respectively. Negative infinity is simply `-KReal::INFINITY`.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 knorba::type::KReal::KReal (const k_real_t *v*)

Constructor; initiates the sotred value with the given argument.

Parameters

v	Initial value.

4.19.3 Member Function Documentation

4.19.3.1 void knorba::type::KReal::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.19.3.2 void knorba::type::KReal::set (const k_real_t *v*) `[virtual]`

Sets the stored value to the one provided.

Parameters

v	The value to assign.

4.19.3.3 void knorba::type::KReal::set (**PPtr**< **KValue** > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.19.3.4 void knorba::type::KReal::writeToBinaryStream (**PPtr**< **OutputStream** > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

4.19.4 Member Data Documentation

4.19.4.1 const double knorba::type::KReal::INFINITY = 0x7FF0000000000000 `[static]`

IEEE 754 representation of positive infinity.

4.19.4.2 const double knorba::type::KReal::NAN = 0x7FFFFFFFFFFFFFFF `[static]`

IEEE 754 representation of not-a-number (NaN).

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

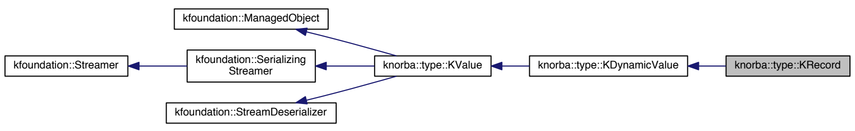
- KReal.h
- KReal.cpp

4.20 knorba::type::KRecord Class Reference

Wrapper class and C++ representation of KnoBRA `record`.

`#include <knorba/type/KRecord.h>`

Inheritance diagram for knorba::type::KRecord:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KDynamicValue
knorba::type::KRecord
kfoundation::StreamDeserializer

Public Member Functions

- KRecord (**PPtr**< KRecordType > type)

  Primary constructor; creates internal storage for a record of the given type.
- KRecord (**PPtr**< KGrid > grid)

  Creates a new record bound to the first cell of the given grid.
- KRecord (**PPtr**< KRecord > record, const k_octet_t fieldIndex)

  Creates a new record bound to the given field of the given record.
- KRecord (**PPtr**< KRecord > record, const string &fieldName)

  Creates a new record bound to the given field of the given record.
- ∼KRecord ()

  Deconstructor.
- **PPtr**< KValue > field (const k_octet_t index) const

  Returns wrapper object for the field at the given index.
- **PPtr**< KValue > field (const string &name) const

  Returns wrapper object for the field with the given name.
- **PPtr**< KString > getString (const k_octet_t index=0) const

  Returns the value of the field at the given index.
- **PPtr**< KString > getString (const string &name) const

  Returns the value of the field at the given name.
- void setString (const k_octet_t index, **PPtr**< KString > value)

  Sets the field at the given index with the value stored in the given wrapper object.
- void setString (const string &name, **PPtr**< KString > value)

  Sets the field with the given name with the value stored in the given wrapper object.
- void setString (**PPtr**< KString > value)

  Sets the first field with the value stored in the given wrapper object.
- void setTruth (const k_octet_t index, const k_truth_t value)

  Sets the field at the given index with the given value.
- void setTruth (const k_truth_t value)

  Sets the first field with the given value.
- void setTruth (const string &name, const k_truth_t value)

  Sets the field with the given name with the given value.
- void setEnumeration (const k_octet_t index, const string &label)

  Sets the value of the enumeration field at the given index with the given label.
- string getEnumerationLabel (const k_octet_t index=0) const

  Returns the label of the enumeration field at the given index.
- k_octet_t getEnumerationOrdinal (const k_octet_t index=0) const

  Returns the ordinal of the enumeration field at the given index.
- **PPtr**< KRecord > getRecord (k_octet_t index) const

  Returns the record at the given index.
- **PPtr**< KRecord > getRecord (const string &name) const

  Returns the record at the field with the given name.
- void getRecord (k_octet_t index, **PPtr**< KRecord > wrapper) const

  Wraps the given record around the field at the given index.
- void getRecord (const string &name, **PPtr**< KRecord > wrapper) const

  Wraps the given record around the field with the given name.
- void setRuntime (Runtime &rt)

  If this record has a field of `any` type, this method should be called before performing `readFromBinaryStream()`.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.

- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Reads the value of this record from the given stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Additional Inherited Members

4.20.1 Detailed Description

Wrapper class and C++ representation of KnoBRA `record`.

A `record` is an ordered sequence of fields of various types.

This class functions in two ways: it can either store values internally, or wrap around a record stored by another KRecord or KGrid object, and provide access to it. See `getRecord(k_octet_t, PPtr<KRecord>)`, `getRecord(const string&, PPtr<KRecord>)`, and `KGrid` for more information.

For user's convenience getXXX() and setXXX() methods are overloaded for every KnoRBA type. Usage:

```
Ptr<KRecordType> dateType = new KRecordType("com.mydomain.Date");
dateType->addField("year", KType::INTEGER)
        ->addField("month", KType::OCTET)
        ->addField("day", KType::OCTET);
Ptr<KRecord> aDate = new KRecord(dateType);
aDate->setInteger("year", 1981);
aDate->setOctet("month", 6);
aDate->setOctet("day", 2);
```

Getter and setters that receive field index instead of field name are faster:

```
aDate->setInteger(0, 1981);
aDate->setOctet(1, 6);
aDate->setOctet(2, 2);
```

Alternatively, a set of method that return type-wrappers can be used. These methods are generally slower than above alternatives:

```
aDate->getInteger("year")->set(1981);
aDate->getOctet("month")->set(6);
aDate->getOctet(2)->set(2);
```

There two small exceptions. For setting enumeration values, there are separate methods for setting by ordinal or by label.

The other exception is when accessing a field of type record.

```
Ptr<KRecordType> logType = new KRecordType("com.mydomain.Log");
logType->addField("date", dateType.AS(KType))
       ->addField("note", KType::STRING);
Ptr<KRecord> log = new KRecord(logType);
```

Then, inner fields can be changes as

```
log->getRecord("date")->setInteger(0, 1981);
```

or, for accessing multiple fields:

```
Ptr<KRecord> date = log->getRecord("date");
date->setInteger("year", 1981);
date->setOctet("month", 6);
date->setOctet("day", 2);
```

Occasionally, it helps to create a wrapper for the inner record in advance:

```
Ptr<KRecord> date = new KRecord(dateType);
log->getRecord("date", date);
cout << *date << endl;
```

Note

> **IMPORTANT.** Field accessor methods do not perform type checking for performance reasons. Take extereme caution to use the correct method. It is very easy to harm internal pointers and other values if a wrong accessor method is used.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 knorba::type::KRecord::KRecord (PPtr< KRecordType > *type*)

Primary constructor; creates internal storage for a record of the given type.

Parameters

type	This record's type.

4.20.2.2 knorba::type::KRecord::KRecord (PPtr< KRecord > *record,* const k_octet_t *fieldIndex*)

Creates a new record bound to the given field of the given record.

The given field should be of a record type.

4.20.2.3 knorba::type::KRecord::KRecord (PPtr< KRecord > *record,* const string & *fieldName*)

Creates a new record bound to the given field of the given record.

The given field should be of a record type.

4.20.2.4 knorba::type::KRecord::∼KRecord ( )

Deconstructor.

Frees any used memory.

4.20.3 Member Function Documentation

4.20.3.1 PPtr< KString > knorba::type::KRecord::getString (const string & *name*) const

Returns the value of the field at the given name.

4.20.3.2 PPtr< KString > knorba::type::KRecord::getString (const k_octet_t *index* = 0) const

Returns the value of the field at the given index.

Index is optional and is assumed 0 if not provided.

4.20.3.3 void knorba::type::KRecord::readFromBinaryStream (PPtr< InputStream > *input*) [virtual]

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.20.3.4 void knorba::type::KRecord::set (PPtr< KValue > *other*) [virtual]

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.20.3.5 void knorba::type::KRecord::setString (PPtr< KString > *value*)

Sets the first field with the value stored in the given wrapper object.

4.20.3.6 void knorba::type::KRecord::setString (const k_octet_t *index*, PPtr< KString > *value*)

Sets the field at the given index with the value stored in the given wrapper object.

4.20.3.7 void knorba::type::KRecord::setString (const string & *name*, PPtr< KString > *value*)

Sets the field with the given name with the value stored in the given wrapper object.

4.20.3.8 void knorba::type::KRecord::setTruth (const string & *name*, const k_truth_t *value*)

Sets the field with the given name with the given value.

4.20.3.9 void knorba::type::KRecord::setTruth (const k_truth_t *value*)

Sets the first field with the given value.

4.20.3.10 void knorba::type::KRecord::setTruth (const k_octet_t *index*, const k_truth_t *value*)

Sets the field at the given index with the given value.

4.20.3.11 void knorba::type::KRecord::writeToBinaryStream (PPtr< OutputStream > *output*) const [virtual]

Reads the value of this record from the given stream.

Note

If this record has fields of type `any`, make sure to call `setRuntime()` before calling this method. Failure to do so will cause an exception to be thrown.

Implements knorba::type::KValue.

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

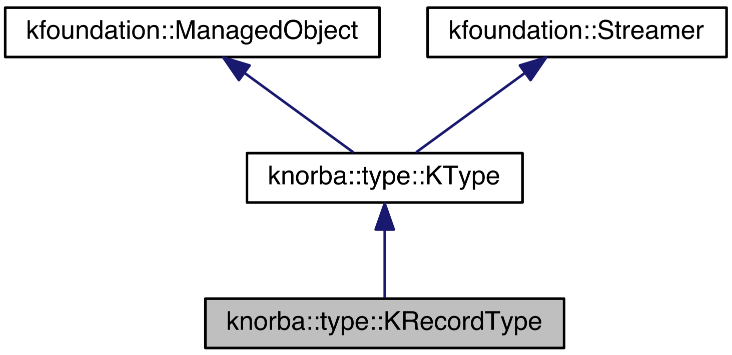
- KRecord.h
- KRecord.cpp

4.21 knorba::type::KRecordType Class Reference

Instantiate to create a custom KnoRBA **record** type.

`#include <knorba/type/KRecordType.h>`

Inheritance diagram for knorba::type::KRecordType:



Public Member Functions

- KRecordType (const string &name)

  Constructor.
- KRecordType (**PPtr**< KType > fieldType)

  Shortcut constructor for records with signle fields.
- **PPtr**< KRecordType > addField (const string &name, **Ptr**< KType > type)

  Adds a new field to record type represented by this object.
- int getNumberOfFields () const

  Returns the number of fields of the record represented by this object.
- string getNameOfFieldAtIndex (const int i) const

  Returns the name of the field at the given index.
- **PPtr**< KType > getTypeOfFieldAtIndex (const int i) const

  Returns the type of the field at the given index.
- **PPtr**< KType > getTypeOfFieldWithName (const string &name) const

  Returns the type of the field with the given name.
- int getIndexForFieldWithName (const string &name) const

  Returns the index of the field with the given name.
- unsigned int getOffsetOfFieldAtIndex (const int index) const

  Memory storage helper method.
- bool hasDynamicFields () const

  Returns `true` iff at least one the fields of the record represented by this object has dynamic length.
- **Ptr**< KGridType > makeGridType (k_octet_t nDims) const

Returns a KGridType with cells of the type represented by this object.

- bool isCastableTo (**PPtr**< KType > t) const

  Checks if the type represented by this object is castable to the given type.
- bool isAutomaticCastableTo (**PPtr**< KType > t) const

  Checks if this type can be automatically casted to the given type by KnoIL language interpreter.
- bool equals (**PPtr**< KType > t) const

  Checks if type represented by this object is equivalant to the one represented by the given argument.
- int getSizeInOctets () const

  If hasConstantSize() returns `true`, this method returns the amount of octets a value of this type consumes when stored in memory or sent over a stream; otherwise it resturns 0.
- bool isPrimitive () const

  Returns `true` iif this object represents a primitive type.
- bool hasConstantSize () const

  Returns true iif the type represented by this object has constant size.
- **Ptr**< KValue > instantiate () const

  Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object.
- string toKnois () const

  Returns type description in KnoIS language.

Additional Inherited Members

4.21.1 Detailed Description

Instantiate to create a custom KnoRBA **record** type.

A record is a collection of fields of various types. Usage:

```
Ptr<KRecordType> dateType = new KRecordType("Date");
myType->addField("year", KType::OCTET)
      ->addField("month", KType::OCTET)
      ->addField("day", KType::OCTET);

Ptr<KRecordType> entryType = new KRecordType("Entry");
myType->addField("name", KType::STRING)
      ->addField("phone", KType::STRING)
      ->addField("birthDay", dateType.AS(KType));
```

A record may have fixed or variable size depending the type of its fields. If hadDynamicFields() returns `true` then, there record has variable size.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 knorba::type::KRecordType::KRecordType (const string & *name*)

Constructor.

Parameters

name	Type name

4.21.2.2 knorba::type::KRecordType::KRecordType (PPtr< KType > *fieldType*)

Shortcut constructor for records with signle fields.

The type name and field name is automatically infered from the type name of the field.

Parameters

fieldType	Type of the record's sole field.

4.21.3 Member Function Documentation

4.21.3.1 PPtr< KRecordType > knorba::type::KRecordType::addField (const string & *name*, Ptr< KType > *type*)

Adds a new field to record type represented by this object.

Parameters

name	Field name.
type	Field type.

Returns

Pointer to self.

4.21.3.2 bool knorba::type::KRecordType::equals (PPtr< KType > *t*) const [virtual]

Checks if type represented by this object is equivalant to the one represented by the given argument.

Checks if this object and the given argument represent the same type.

Reimplemented from knorba::type::KType.

4.21.3.3 int knorba::type::KRecordType::getIndexForFieldWithName (const string & *name*) const

Returns the index of the field with the given name.

Returns -1 if there is no such field.

Parameters

name	The name of the field to retrieve index of.

4.21.3.4 string knorba::type::KRecordType::getNameOfFieldAtIndex (const int *i*) const

Returns the name of the field at the given index.

Parameters

i	An index between 0 and getNumberOfFields() - 1.

4.21.3.5 unsigned int knorba::type::KRecordType::getOffsetOfFieldAtIndex (const int *index*) const

Memory storage helper method.

Returns the offset at which the field with the given index is stored, calculated from the point at which the first field is stored.

Parameters

index	An Index between 0 to getNumberOfFields() - 1.

4.21.3.6 PPtr< KType > knorba::type::KRecordType::getTypeOfFieldAtIndex (const int *i*) const

Returns the type of the field at the given index.

Parameters

i	An index between 0 and getNumberOfFields() - 1.

4.21.3.7 PPtr< KType > knorba::type::KRecordType::getTypeOfFieldWithName (const string & *name*) const

Returns the type of the field with the given name.

Returns a null pointer if such field does not exist.

Parameters

name	The name of the field to retrieve type for.

4.21.3.8 bool knorba::type::KRecordType::hasConstantSize ( ) const [virtual]

Returns true iif the type represented by this object has constant size.

Types with variable size are `string` (KType::STRING), `raw` (KType::RAW), and `grid` (KGridType). The rest of them have constant sizes.

Implements knorba::type::KType.

4.21.3.9 bool knorba::type::KRecordType::hasDynamicFields ( ) const

Returns `true` iff at least one the fields of the record represented by this object has dynamic length.

That is, at least one of the fields is is of type `string`, `raw`, `grid`, or a `record` for which hasDynamicFields() returns `true`.

4.21.3.10 Ptr< KValue > knorba::type::KRecordType::instantiate ( ) const [virtual]

Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object. For example,

```
KType::INTEGER->instantiate()
```

will return an instance of KInteger.

Implements knorba::type::KType.

4.21.3.11 Ptr< KGridType > knorba::type::KRecordType::makeGridType (k_octet_t *nDims*) const

Returns a KGridType with cells of the type represented by this object.

Parameters

nDims	The number of dimensions of the resulting grid type.

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

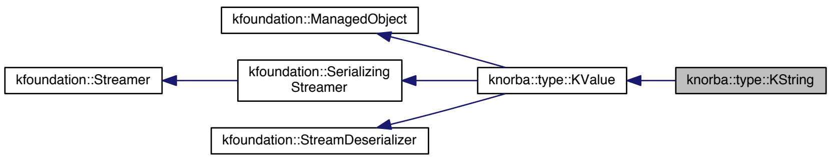
- KRecordType.h
- KRecordType.cpp

4.22 knorba::type::KString Class Reference

Wrapper class and C++ representation of KnoRBA `string` type.

`#include <knorba/type/KString.h>`

Inheritance diagram for knorba::type::KString:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KString
kfoundation::StreamDeserializer

Public Member Functions

- KString ()

  Constructor; creates an empty KnoRBA string.
- KString (const string &str)

  Constructor; copies the stored value from the given string.
- KString (const wstring &str)

  Constructor; copies the stored value from the given string.
- ∼KString ()

  Deconstructor.
- k_longint_t getHashCode () const

  Returns the hashcode of the stored string (64-bit CityHash).
- k_longint_t getNOctets () const

  Returns the number of octets in the stored string.
- void set (const string &str)

  Sets the stored value from the given string.
- void set (const wstring &str)

  Sets the stored value from the given string.
- k_longint_t getNCodePoints () const

  Returns the number of code points (characters) in this string.
- wstring toWString () const

  Converts the stored string into C++ wstring.
- const char ∗ getUtf8CStr () const

  Returns the pointer to the internal buffer where UTF-8 encoded string is stored.
- string toUtf8String () const

  Creates a new std::string object containing the same UTF-8 representation as this object.
- wchar_t getCodePointAt (const k_longint_t index) const

Returns codepoint (character) at the given index.

- k_octet_t getOctetAt (const k_longint_t index) const

  Returns the octet at the given index.
- bool equals (const wstring &ws) const

  Checks if this object and the given std::wstring object represent the same string.
- bool equals (const string &s) const

  Checks if this object and the UTF-8 encoded std::string object represent the same string.
- bool equals (**PPtr**< KString > str) const

  Checks if string stored in this object is equal to the one stored by the given parameter.
- bool hashEquals (const k_longint_t &hash) const

  Checks if the hashcode of this string is equal to the given value.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.
- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Member Functions

- static k_longint_t generateHashFor (const wstring &ws)

  Generates 64-bit CityHash hashcode for the given string.
- static k_longint_t generateHashFor (const string &s)

  Generates 64-bit CityHash hashcode for the given string.
- static k_longint_t generateHashFor (const k_octet_t ∗s, k_longint_t size)

  Generates 64-bit CityHash hashcode for the given sequence of octets.

Additional Inherited Members

4.22.1 Detailed Description

Wrapper class and C++ representation of KnoRBA `string` type.

KnoRBA `string`s are encoded in UTF-8.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 knorba::type::KString::KString (const string & *s*)

Constructor; copies the stored value from the given string.

Parameters

s	Initial value.

4.22.2.2 knorba::type::KString::KString (const wstring & *str*)

Constructor; copies the stored value from the given string.

Parameters

str	Initial value.

4.22.2.3 knorba::type::KString::∼KString ( )

Deconstructor.

Deletes the internal buffer.

4.22.3 Member Function Documentation

4.22.3.1 bool knorba::type::KString::equals (const wstring & *ws*) const

Checks if this object and the given std::wstring object represent the same string.

This method works by comparing hashcodes.

4.22.3.2 bool knorba::type::KString::equals (const string & *s*) const

Checks if this object and the UTF-8 encoded std::string object represent the same string.

This method works by comparing hashcode.

4.22.3.3 k_longint_t knorba::type::KString::generateHashFor (const k_octet_t ∗ *s*, k_longint_t *size*) `[static]`

Generates 64-bit CityHash hashcode for the given sequence of octets.

Parameters

s	Pointer to the begining of the sequence.
size	The number of octets in the sequence.

4.22.3.4 void knorba::type::KString::readFromBinaryStream (PPtr< InputStream > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.22.3.5 void knorba::type::KString::set (PPtr< KValue > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.22.3.6 void knorba::type::KString::writeToBinaryStream (**PPtr**< **OutputStream** > *output*) const `[virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

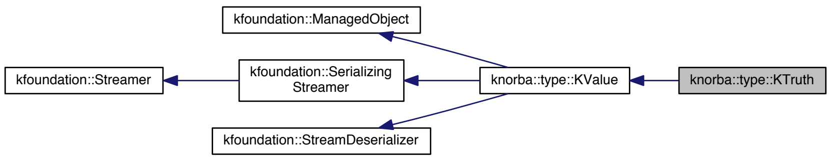
- KString.h
- KString.cpp

4.23 knorba::type::KTruth Class Reference

Wrapper class for KnoRBA 3-state `truth` type.

`#include <knorba/type/KTruth.h>`

Inheritance diagram for knorba::type::KTruth:


kfoundation::ManagedObject
kfoundation::Streamer
kfoundation::Serializing Streamer
knorba::type::KValue
knorba::type::KTruth
kfoundation::StreamDeserializer

Public Member Functions

- KTruth ()

  Constructor; sets the stored value to X.
- KTruth (const k_truth_t v)

  Constructor; sets the stored value to the given argument.
- virtual void set (const k_truth_t v)

  Sets the stored value to the given argument.
- virtual k_truth_t get () const

  Returns the stored value.
- void set (**PPtr**< KValue > other)

  Copies the value for this KValue from another one.
- **PPtr**< KType > getType () const

  Returns the KnoRBA type for the stored value.
- k_longint_t getTotalSizeInOctets () const

  Returns the size of the stored value when serialized.
- void readFromBinaryStream (**PPtr**< **InputStream** > input)

  Sets the stored value by deserializing the given input stream.

- void writeToBinaryStream (**PPtr**< **OutputStream** > output) const

  Serializes the stored value on to the given output stream.
- void deserialize (**PPtr**< **ObjectToken** > headToken)

  Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- void serialize (**PPtr**< **ObjectSerializer** > builder) const

  Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Member Functions

- static string toString (const k_truth_t v)

  Returns string representation of the given scalar `truth` *value.*

Additional Inherited Members

4.23.1 Detailed Description

Wrapper class for KnoRBA 3-state `truth` type.

A value of type `truth` can be either `T` (for true), `F` (for false), or `X` (for unknown). These values are represented by `knorba::type::T`, `knorba::type::F`, and `knorba::type::X`, respectively. The scalar type associated with this wrapper class is `knorba::type::k_truth_t`.

KnoRBA employs 3-state logic rather than the common Boolean logic mostly because it simplifies implementation of KnoIL interpreter and makes it easier for a group of agents to engage in collaborative descision making.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 knorba::type::KTruth::KTruth (const k_truth_t *v*)

Constructor; sets the stored value to the given argument.

Parameters

v	Initial value.

4.23.3 Member Function Documentation

4.23.3.1 void knorba::type::KTruth::readFromBinaryStream (**PPtr**< **InputStream** > *input*) `[virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implements knorba::type::KValue.

4.23.3.2 void knorba::type::KTruth::set (**PPtr**< **KValue** > *other*) `[virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implements knorba::type::KValue.

4.23.3.3 void knorba::type::KTruth::writeToBinaryStream (**PPtr**< **OutputStream** > *output*) const [virtual]

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implements knorba::type::KValue.

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

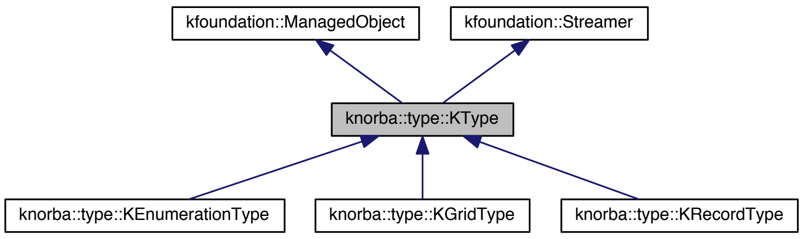
- KTruth.h
- KTruth.cpp

4.24 knorba::type::KType Class Reference

Represents a KnoRBA type, and offers useful runtime information about them.

```
#include <knorba/type/KType.h>
```

Inheritance diagram for knorba::type::KType:


kfoundation::ManagedObject
kfoundation::Streamer
knorba::type::KType
knorba::type::KEnumerationType
knorba::type::KGridType
knorba::type::KRecordType

Public Member Functions

- const string & getTypeName () const

  Returns the type name.
- k_longint_t getTypeNameHash () const

  Returns the hashcode for type name (64-bit CityHash)
- virtual bool isCastableTo (**PPtr**< KType > t) const =0

  Checks if the type represented by this object is castable to the given type.
- virtual bool isAutomaticCastableTo (**PPtr**< KType > t) const =0

  Checks if this type can be automatically casted to the given type by KnoIL language interpreter.
- virtual bool equals (**PPtr**< KType > t) const

  Checks if type represented by this object is equivalant to the one represented by the given argument.
- virtual int getSizeInOctets () const =0

  If hasConstantSize() returns `true`, this method returns the amount of octets a value of this type consumes when stored in memory or sent over a stream; otherwise it resturns 0.
- virtual bool isPrimitive () const =0

  Returns `true` iif this object represents a primitive type.
- virtual bool hasConstantSize () const =0

  Returns true iif the type represented by this object has constant size.
- virtual **Ptr**< KValue > instantiate () const =0

  Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object.
- virtual string toKnois () const

  Returns type description in KnoIS language.

- void printToStream (ostream &os) const

    Implements compatibility with **kfoundation::Streamer** *interface.*

Static Public Attributes

- static const **SPtr**< KType > TRUTH

    Runtime representation of KnoRBA ***truth*** *type.*
- static const **SPtr**< KType > OCTET

    Runtime representation of KnoRBA ***octet*** *type.*
- static const **SPtr**< KType > INTEGER

    Runtime representation of KnoRBA ***integer*** *type.*
- static const **SPtr**< KType > LONGINT

    Runtime representation of KnoRBA ***longint*** *type.*
- static const **SPtr**< KType > REAL

    Runtime representation of KnoRBA ***real*** *type.*
- static const **SPtr**< KType > GUID

    Runtime representation of KnoRBA ***GUID*** *type.*
- static const **SPtr**< KType > STRING

    Runtime representation of KnoRBA ***string*** *type.*
- static const **SPtr**< KType > RAW

    Runtime representation of KnoRBA ***raw*** *type.*
- static const **SPtr**< KType > ANY

    Runtime representation of KnoRBA ***any*** *type.*
- static const **SPtr**< KType > NOTHING

    Runtime representation of KnoRBA ***nothing*** *type.*

Protected Member Functions

- KType (string name)

    Constructor.

4.24.1 Detailed Description

Represents a KnoRBA type, and offers useful runtime information about them.

KType and its derivatives interact with KValue and its derivatives as follows.
Type to value:

```
Ptr<KValue> value = type->instantiate();
```

Value to type:

```
Ptr<KType> type = value->getType();
```

KnoRBA primitive types are represented by constant memebers of KType: TRUTH, OCTET, INTEGER, LONGINT, REAL, GUID, STRING, RAW, ANY, and NOTHING.

Compound types are represented by KType derivatives, KEnumerationType, KRecordType and KGridType.

Note

In case of types for which hasConstantSize() returns `false`, getSizeInOctets() always returns 0. You need to invoke KValue::getTotalSizeInOctets() on each instance to determine their size.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 knorba::type::KType::KType (string *name*) `[protected]`

Constructor.

Parameters

name	Type name

4.24.3 Member Function Documentation

4.24.3.1 bool knorba::type::KType::equals (**PPtr< KType >** *t*) const [virtual]

Checks if type represented by this object is equivalant to the one represented by the given argument.

Checks if this object and the given argument represent the same type.

Reimplemented in knorba::type::KEnumerationType, knorba::type::KRecordType, and knorba::type::KGridType.

4.24.3.2 virtual bool knorba::type::KType::hasConstantSize ( ) const [pure virtual]

Returns true iif the type represented by this object has constant size.

Types with variable size are `string` (KType::STRING), `raw` (KType::RAW), and `grid` (KGridType). The rest of them have constant sizes.

Implemented in knorba::type::KEnumerationType, knorba::type::KRecordType, and knorba::type::KGridType.

4.24.3.3 virtual Ptr<KValue> knorba::type::KType::instantiate ( ) const [pure virtual]

Returns an instance of an appropriate subclass of KValue corresponding to the type represented by this object. For example,

```
KType::INTEGER->instantiate()
```

will return an instance of KInteger.

Implemented in knorba::type::KEnumerationType, knorba::type::KRecordType, and knorba::type::KGridType.

4.24.3.4 void knorba::type::KType::printToStream (ostream & *os*) const

Implements compatibility with **kfoundation::Streamer** interface.

Internally uses the output of toKnois() methos.

4.24.4 Member Data Documentation

4.24.4.1 const SPtr< KType > knorba::type::KType::ANY [static]

Runtime representation of KnoRBA **any** type.

4.24.4.2 const SPtr< KType > knorba::type::KType::GUID [static]

Runtime representation of KnoRBA **GUID** type.

4.24.4.3 const SPtr< KType > knorba::type::KType::INTEGER [static]

Runtime representation of KnoRBA **integer** type.

4.24.4.4 const SPtr< KType > knorba::type::KType::LONGINT `[static]`

Runtime representation of KnoRBA **longint** type.

4.24.4.5 const SPtr< KType > knorba::type::KType::NOTHING `[static]`

Runtime representation of KnoRBA **nothing** type.

4.24.4.6 const SPtr< KType > knorba::type::KType::OCTET `[static]`

Runtime representation of KnoRBA **octet** type.

4.24.4.7 const SPtr< KType > knorba::type::KType::RAW `[static]`

Runtime representation of KnoRBA **raw** type.

4.24.4.8 const SPtr< KType > knorba::type::KType::REAL `[static]`

Runtime representation of KnoRBA **real** type.

4.24.4.9 const SPtr< KType > knorba::type::KType::STRING `[static]`

Runtime representation of KnoRBA **string** type.

4.24.4.10 const SPtr< KType > knorba::type::KType::TRUTH `[static]`

Runtime representation of KnoRBA **truth** type.

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

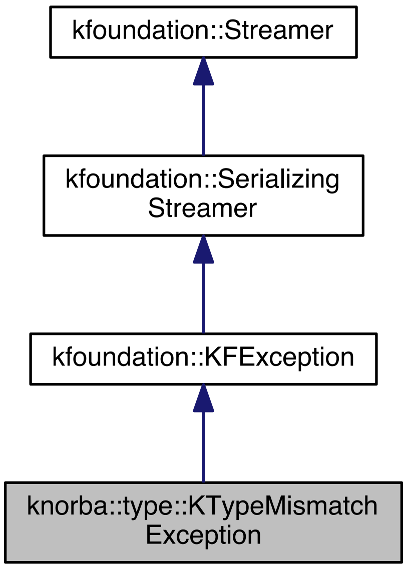
- KType.h
- KType.cpp

4.25 knorba::type::KTypeMismatchException Class Reference

Exception indicating mismatch of two KnoRBA types.

```
#include <knorba/type/KTypeMismatchException.h>
```

Inheritance diagram for knorba::type::KTypeMismatchException:


kfoundation::Streamer
kfoundation::Serializing Streamer
kfoundation::KFException
knorba::type::KTypeMismatch Exception

Public Member Functions

- KTypeMismatchException (**PPtr**< KType > expected, **PPtr**< KType > provided)

  Constructor.

4.25.1 Detailed Description

Exception indicating mismatch of two KnoRBA types.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 knorba::type::KTypeMismatchException::KTypeMismatchException (PPtr< KType > *expected,* PPtr< KType > *provided*)

Constructor.

Parameters

expected	The expected type.
privided	The provided type.

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

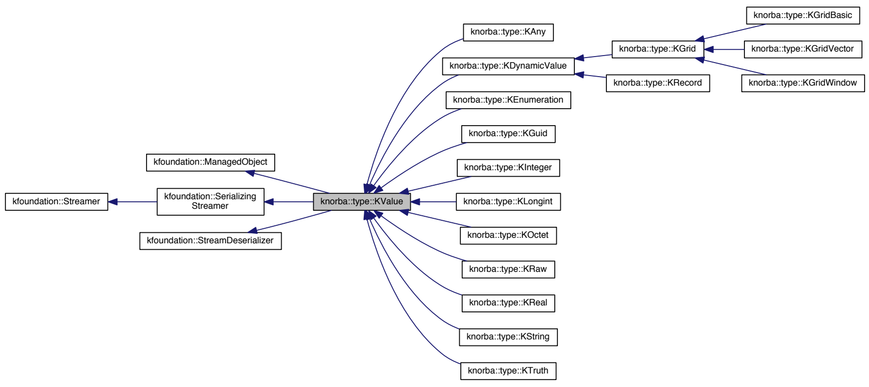
- KTypeMismatchException.h
- KTypeMismatchException.cpp

4.26 knorba::type::KValue Class Reference

Abstract superclass for all KnoRBA type-wrapper classes.

`#include <knorba/type/KValue.h>`

Inheritance diagram for knorba::type::KValue:



Public Member Functions

- virtual void set (**PPtr**< KValue > other)=0

    Copies the value for this KValue from another one.
- virtual **PPtr**< KType > getType () const =0

    Returns the KnoRBA type for the stored value.
- virtual k_longint_t getTotalSizeInOctets () const =0

    Returns the size of the stored value when serialized.
- virtual void readFromBinaryStream (**PPtr**< **InputStream** > input)=0

    Sets the stored value by deserializing the given input stream.
- virtual void writeToBinaryStream (**PPtr**< **OutputStream** > output) const =0

    Serializes the stored value on to the given output stream.
- virtual void deserialize (**PPtr**< **ObjectToken** > headToken)=0

    Implements compatibility with **kfoundation::StreamDeserializer** *interface.*
- virtual void serialize (**PPtr**< **ObjectSerializer** > serializer) const =0

    Implements compatibility with **kfoundation::SerializingStreamer** *interface.*

Static Public Attributes

- static const **SPtr**< KValue > NOTHING

    Wrapper for KnoRBA `nothing` *literal.*

4.26.1 Detailed Description

Abstract superclass for all KnoRBA type-wrapper classes.

Wrapper classes are responsible for storing, managing, serializing, and deseralizaing KnoRBA binary data format.

4.26.2 Member Function Documentation

4.26.2.1 virtual void knorba::type::KValue::readFromBinaryStream (**PPtr**< **InputStream** > *input*) `[pure virtual]`

Sets the stored value by deserializing the given input stream.

Parameters

input	The input stream to deserialize from.

Implemented in knorba::type::KRecord, knorba::type::KGrid, knorba::type::KString, knorba::type::KGuid, knorba::type::KRaw, knorba::type::KTruth, knorba::type::KReal, knorba::type::KEnumeration, knorba::type::KAny, knorba::type::KInteger, knorba::type::KLongint, and knorba::type::KOctet.

4.26.2.2 virtual void knorba::type::KValue::set (**PPtr< KValue >** *other*) `[pure virtual]`

Copies the value for this KValue from another one.

The given KValue should be of the same type as this one. I.e. `this->getType()->equals(other->getType())` shoule return `true`.

Implemented in knorba::type::KRecord, knorba::type::KGrid, knorba::type::KString, knorba::type::KGuid, knorba::type::KRaw, knorba::type::KTruth, knorba::type::KInteger, knorba::type::KReal, knorba::type::KEnumeration, knorba::type::KOctet, knorba::type::KAny, and knorba::type::KLongint.

4.26.2.3 virtual void knorba::type::KValue::writeToBinaryStream (**PPtr< OutputStream >** *output*) const `[pure virtual]`

Serializes the stored value on to the given output stream.

Parameters

output	The output stream to serialize to.

Implemented in knorba::type::KRecord, knorba::type::KGrid, knorba::type::KString, knorba::type::KGuid, knorba::type::KRaw, knorba::type::KTruth, knorba::type::KReal, knorba::type::KEnumeration, knorba::type::KAny, knorba::type::KInteger, knorba::type::KLongint, and knorba::type::KOctet.

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

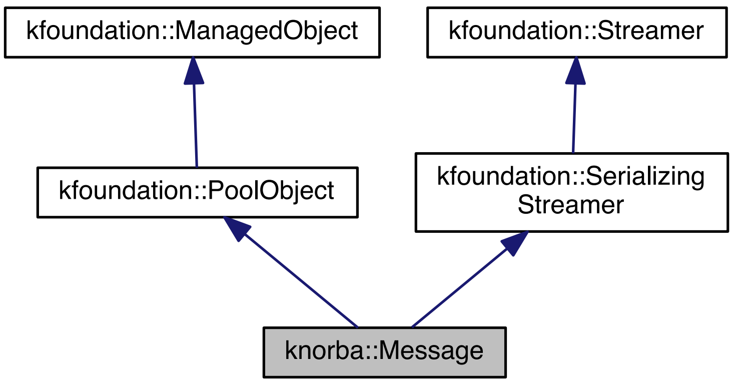
- KValue.h
- KValue.cpp

4.27 knorba::Message Class Reference

Represents a KnoRBA message.

`#include <knorba/Message.h>`

Inheritance diagram for knorba::Message:



Public Member Functions

- void set (const k_integer_t tid, const k_longint_t opcodeHash, const k_guid_t &sender, **PPtr**< KValue > _payload)

  Setter.
- k_integer_t getTransactionId () const

  Returns transaction ID.
- k_longint_t getOpcodeHash () const

  Returns Opcode Hash.
- const k_guid_t & getSender () const

  Returns the sender's GUID.
- **PPtr**< KValue > getPayload () const

  Returns the message payload.
- bool is (**PPtr**< KString > opcode) const

  Checks if the opcode of this message matches the given string.
- bool needsResponse () const

  Checks if this message is blocking the sender for a response.
- string headerToString (Runtime &rt) const

  Converts header information to string.

4.27.1 Detailed Description

Represents a KnoRBA message.

Note

KnoRBA messages carry opcode hash instead of opcode itself. If ever needed, use is() method to check the opcode.
This is a pool-allocated object and should not be instanitated directly.

4.27.2 Member Function Documentation

4.27.2.1 k_integer_t knorba::Message::getTransactionId ( ) const

Returns transaction ID.

4.27.2.2 bool knorba::Message::needsResponse ( ) const

Checks if this message is blocking the sender for a response.

If returns true, use Agent::respond() to respond and use this message as the first argument.

4.27.2.3 void knorba::Message::set (const k_integer_t *tid*, const k_longint_t *opcodeHash*, const **k_guid_t** & *sender*, **PPtr**< **KValue** > *payload*)

Setter.

Replaces all values of this object with the ones given in arguments.

Parameters

tid	Transaction ID.
opcodeHash	Opcode Hash (64-bit CityHash).
sender	GUID of the sender agent.
payload	Payload.

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

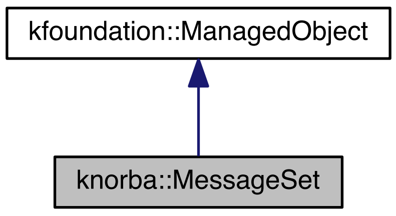
- Message.h
- Message.cpp

4.28 knorba::MessageSet Class Reference

Container for a collection of messages.

`#include <knorba/MessageSet.h>`

Inheritance diagram for knorba::MessageSet:



Public Member Functions

- MessageSet ()

Sole constructor.
- void add (**Ptr**< Message > msg)

  Adds a message to this set.
- int getSize () const

  Returns the number of message in this set.
- **PPtr**< Message > get (int index) const

  Returns the message at the given index.
- **PPtr**< Group > getSenders () const

  Returns the GUIDs of all senders of messages in this set.
- bool isEmpty () const

  Checks if this set is empty.
- void clear ()

  Removes all messages in this set.

4.28.1 Detailed Description

Container for a collection of messages.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 knorba::MessageSet::MessageSet ( )

Sole constructor.

4.28.3 Member Function Documentation

4.28.3.1 void knorba::MessageSet::add (Ptr< Message > *msg*)

Adds a message to this set.

4.28.3.2 PPtr< Message > knorba::MessageSet::get (int *index*) const

Returns the message at the given index.

4.28.3.3 int knorba::MessageSet::getSize ( ) const

Returns the number of message in this set.

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

- MessageSet.h
- MessageSet.cpp

4.29 knorba::Protocol Class Reference

Protocols are the way code reusability in KnoRBA is achieved.

`#include <knorba/Protocol.h>`

Inherited by knorba::protocol::ACellProtocol, knorba::protocol::ConsoleProtocolClient, knorba::protocol::DisplayInfoProtocol, knorba::protocol::GroupingProtocol, knorba::protocol::PhaserProtocol, knorba::protocol::TunnelingProtocol, and knorba::protocol::UnixSocketClient.

Public Types

- typedef void(Protocol::∗ phandler_t)(**PPtr**< Message >)

  Pointer to protocol message handler.

Public Member Functions

- Protocol (Agent ∗_owner)

  Constructor.
- ∼Protocol ()

  Deconstructor.
- virtual void handlePeerConnectionReuqest (**PPtr**< KString > role, const k_guid_t &guid)

  Override to handle peer connection request.
- virtual void handlePeerDisconnected (**PPtr**< KString > role, const k_guid_t &guid)

  Override to handle peer disconnect notifications.
- virtual void finalize ()

  Override to perform additional tasks when agent using this protocol is finalizing.
- virtual bool isAlive () const

  Override if there are additional criteria to determine this agent is alive.

Protected Member Functions

- void registerHandler (phandler_t handler, **PPtr**< KString > opcode)

  Registers a handler for the given opcode.

4.29.1 Detailed Description

Protocols are the way code reusability in KnoRBA is achieved.

A protocol implements a particular behavior, and all agents using that protocol will inherite that behavior. In a way, protocols are realization of horizontal inheritence – in contrast with vertical inheritence in Object-Oriented programming.

Derive this class to define a custom protocol.

Just as in Agent class, a Protocol declares a set of message handlers each corresponding to a given opcode. A Protocol can also define roles and reactions to additiona and removal of peers from those roles.

Usage:

```
MyAgent::MyAgent(Runtime& rt, const k_guid_t& guid)
: Agent(rt, guid),
  myProtocol(this)
{
    ... constructor code ...
}
```

4.29.2 Member Function Documentation

4.29.2.1 void knorba::Protocol::finalize ( ) [virtual]

Override to perform additional tasks when agent using this protocol is finalizing.

Stops the message processor thread.

See Also

isAlive()
Agent::finalize()

4.29.2.2 knorba::Protocol::handlePeerConnectionReuqest (PPtr< KString > *role*, const k_guid_t & *guid*) [virtual]

Override to handle peer connection request.

Parameters

role	The request role for the new peer.
guid	The GUID of the agent requesting to become a peer.

See Also

handlePeerDisconnected()
Agent::handlePeerConnectionRequest()

4.29.2.3 void knorba::Protocol::handlePeerDisconnected (PPtr< KString > *role*, const k_guid_t & *guid*) [virtual]

Override to handle peer disconnect notifications.

Parameters

role	The role of the peer to be removed.
guid	The GUID of the agent requesting to be removed as peer.

See Also

handlePeerConnectionReuqest()
Agent::handlePeerDisconnected()

4.29.2.4 bool knorba::Protocol::isAlive ( ) const [virtual]

Override if there are additional criteria to determine this agent is alive.

E.g. other threads are running, connections are open, etc.

See Also

finalize()
Agent::isAlive()

4.29.2.5 void knorba::Protocol::registerHandler (phandler_t *handler*, PPtr< KString > *opcode*) [protected]

Registers a handler for the given opcode.

Parameters

handler	Pointer to handler method
opcode	The opcode that activates the given handler

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

- Protocol.h
- Protocol.cpp

4.30 knorba::Runtime Class Reference

ARE access interface.

```
#include <knorba/Runtime.h>
```

Public Member Functions

- virtual const k_guid_t & getGuid () const =0

    Returns the GUID of ARE.
- virtual void registerType (**PPtr**< KType > type)=0

    Adds a new type to the ARE's type table.
- virtual **PPtr**< KType > getTypeByHash (const k_longint_t hash) const =0

    Find a registered type by its hashcode.
- virtual const k_guid_t & getConsoleGuid () const =0

    Returns the GUID of the default ConsoleAgent assigned to this ARE.
- virtual const string & getAppName () const =0

    Returns the name the current application.
- virtual k_integer_t getNodeCount () const =0

    Returns the number of nodes of the cluster this ARE is a part of.
- virtual bool isHead () const =0

    Returns true if this ARE is the head of its cluster.
- virtual void registerMessageFormat (**PPtr**< KString > opcode, **PPtr**< KType > payloadType)=0

    Registers a new record in ARE's message type table.
- virtual **PPtr**< KType > getMessageFormatByHash (const k_longint_t hash) const =0

    Returns the message type for the given opcode hash.
- virtual **PPtr**< KString > getMessageOpCodeForHash (const k_longint_t hash) const =0

    Returns the opcode for the given opcode hash.

4.30.1 Detailed Description

ARE access interface.

4.30.2 Member Function Documentation

4.30.2.1 virtual const string& knorba::Runtime::getAppName ( ) const `[pure virtual]`

Returns the name the current application.

This must be the same as the name of corresponding KAR or KAP file.

4.30.2.2 virtual const k_guid_t& knorba::Runtime::getGuid ( ) const `[pure virtual]`

Returns the GUID of ARE.

4.30.2.3 virtual PPtr<KType> knorba::Runtime::getMessageFormatByHash (const k_longint_t *hash*) const [pure virtual]

Returns the message type for the given opcode hash.

Parameters

hash	The hash code for the opcode to be looked up (64-bit CityHash).

Returns

The type associated with the given code, or null if such type does not exist.

4.30.2.4 virtual PPtr<KString> knorba::Runtime::getMessageOpCodeForHash (const k_longint_t *hash*) const `[pure virtual]`

Returns the opcode for the given opcode hash.

Parameters

hash	The hash code for the opcode to be looked up (64-bit CityHash).

Returns

The opcode associated with the given hash, or null if such opcode does not exist.

4.30.2.5 virtual PPtr<KType> knorba::Runtime::getTypeByHash (const k_longint_t *hash*) const `[pure virtual]`

Find a registered type by its hashcode.

Parameters

hash	Hashcode for a type name (64-bit CityHash)

Returns

The type with the given name hash, or `null` if such type does not exist.

4.30.2.6 virtual void knorba::Runtime::registerMessageFormat (PPtr< KString > *opcode*, PPtr< KType > *payloadType*) `[pure virtual]`

Registers a new record in ARE's message type table.

Parameters

opcode	The opcode for the message type.
payloadType	The type associated with the given opcode.

4.30.2.7 virtual void knorba::Runtime::registerType (PPtr< KType > *type*) `[pure virtual]`

Adds a new type to the ARE's type table.

Parameters

type	The type to register.

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

- Runtime.h

Index