NAME

Net::DBus::Binding::Message − Base class for messages

SYNOPSIS

Sending a message

```
  my $msg = new Net::DBus::Binding::Message::Signal;
  my $iterator = $msg->iterator;

  $iterator->append_byte(132);
  $iterator->append_int32(14241);

  $connection->send($msg);
```

DESCRIPTION

Provides a base class for the different kinds of message that can be sent/received. Instances of this class are never instantiated directly, rather one of the four sub-types Net::DBus::Binding::Message::Signal, Net::DBus::Binding::Message::MethodCall, Net::DBus::Binding::Message::MethodReturn, Net::DBus::Binding::Message::Error should be used.

CONSTANTS

The following constants are defined in this module. They are not exported into the caller's namespace & thus must be referenced with their fully qualified package names

TYPE_ARRAY
: Constant representing the signature value associated with the array data type.

TYPE_BOOLEAN
: Constant representing the signature value associated with the boolean data type.

TYPE_BYTE
: Constant representing the signature value associated with the byte data type.

TYPE_DICT_ENTRY
: Constant representing the signature value associated with the dictionary entry data type.

TYPE_DOUBLE
: Constant representing the signature value associated with the IEEE double precision floating point data type.

TYPE_INT16
: Constant representing the signature value associated with the signed 16 bit integer data type.

TYPE_INT32
: Constant representing the signature value associated with the signed 32 bit integer data type.

TYPE_INT64
: Constant representing the signature value associated with the signed 64 bit integer data type.

TYPE_OBJECT_PATH
: Constant representing the signature value associated with the object path data type.

TYPE_STRING
: Constant representing the signature value associated with the UTF−8 string data type.

TYPE_SIGNATURE
: Constant representing the signature value associated with the signature data type.

TYPE_STRUCT
: Constant representing the signature value associated with the struct data type.

TYPE_UINT16
: Constant representing the signature value associated with the unsigned 16 bit integer data type.

TYPE_UINT32
: Constant representing the signature value associated with the unsigned 32 bit integer data type.

TYPE_UINT64
: Constant representing the signature value associated with the unsigned 64 bit integer data type.

TYPE_VARIANT
: Constant representing the signature value associated with the variant data type.

TYPE_UNIX_FD
: Constant representing the signature value associated with the unix file descriptor data type.

METHODS

my `$msg` = Net::DBus::Binding::Message–>new(message => `$rawmessage`);
: Creates a new message object, initializing it with the underlying C message object given by the `message` object. This constructor is intended for internal use only, instead refer to one of the four sub-types for this class for specific message types

my `$type` = `$msg`–>get_type
: Retrieves the type code for this message. The returned value corresponds to one of the four `Net::DBus::Binding::Message::MESSAGE_TYPE_*` constants.

my `$interface` = `$msg`–>get_interface
: Retrieves the name of the interface targeted by this message, possibly an empty string if there is no applicable interface for this message.

my `$path` = `$msg`–>get_path
: Retrieves the object path associated with the message, possibly an empty string if there is no applicable object for this message.

my `$name` = `$msg`–>get_destination
: Retrieves the unique or well-known bus name for client intended to be the recipient of the message. Possibly returns an empty string if the message is being broadcast to all clients.

my `$name` = `$msg`–>get_sender
: Retireves the unique name of the client sending the message

my `$serial` = `$msg`–>get_serial
: Retrieves the unique serial number of this message. The number is guaranteed unique for as long as the connection over which the message was sent remains open. May return zero, if the message is yet to be sent.

my `$name` = `$msg`–>get_member
: For method calls, retrieves the name of the method to be invoked, while for signals, retrieves the name of the signal.

my `$sig` = `$msg`–>get_signature
: Retrieves a string representing the type signature of the values packed into the body of the message.

`$msg`–>set_sender($name)
: Set the name of the client sending the message. The name must be the unique name of the client.

`$msg`–>set_destination($name)
: Set the name of the intended recipient of the message. This is typically used for signals to switch them from broadcast to unicast.

my `$iterator` = `$msg`–>iterator;
: Retrieves an iterator which can be used for reading or writing fields of the message. The returned object is an instance of the `Net::DBus::Binding::Iterator` class.

`$boolean` = `$msg`–>**get_no_reply()**
: Gets the flag indicating whether the message is expecting a reply to be sent.

`$msg->set_no_reply($boolean)`
: Toggles the flag indicating whether the message is expecting a reply to be sent. All method call messages expect a reply by default. By toggling this flag the communication latency is reduced by removing the need for the client to wait

`my @values = $msg->get_args_list`
: De-marshall all the values in the body of the message, using the message signature to identify data types. The values are returned as a list.

`$msg->append_args_list(@values)`
: Append a set of values to the body of the message. Values will be encoded as either a string, list or dictionary as appropriate to their Perl data type. For more specific data typing needs, the Net::DBus::Binding::Iterator object should be used instead.

AUTHOR

Daniel P. Berrange

COPYRIGHT

SEE ALSO

Net::DBus::Binding::Server, Net::DBus::Binding::Connection, Net::DBus::Binding::Message::Signal, Net::DBus::Binding::Message::MethodCall, Net::DBus::Binding::Message::MethodReturn, Net::DBus::Binding::Message::Error