RabbitMQ .NET client library API guide

Table of Contents

Table of Contents

Table of Contents

RabbitMQ .NET client library API guide

Copyright

License

Master Index

Namespaces

Namespace	Summary
RabbitMQ.Client	Main public API to the RabbitMQ .NET AMQP client library.
RabbitMQ.Client.Content	Public API for construction and analysis of messages that are binary-compatible with messages produced and consumed by QPid's JMS compatibility layer.
RabbitMQ.Client.Events	Public API for various events and event handlers that are part of the AMQP client library.
RabbitMQ.Client.Exceptions	Public API for exceptions visible to the user of the AMQP client library.
RabbitMQ.Client.MessagePatterns	Public API for high-level helper classes and interface for common ways of using the AMQP client library.
RabbitMQ.Util	Internal. Utility classes.

Types

Type	Summary
RabbitMQ.Client.AmqpTcpEndpoint	Represents a TCP-addressable AMQ peer, including the protocol variant use, and a host name and port number.
RabbitMQ.Client.AmqpTimestamp	Structure holding an AMQP timestamp, a posix 64-bit time_t.
RabbitMQ.Client.AmqpVersion	Represents a version of the AMQP specification.
RabbitMQ.Client.AuthMechanism	A pluggable authentication mechanism.
RabbitMQ.Client.AuthMechanismFactory	(undocumented)
RabbitMQ.Client.BasicGetResult	Represents Basic.GetOk responses from the server.
RabbitMQ.Client.BinaryTableValue	Wrapper for a byte[]. May appear as values read from and written to AMQP field tables.
RabbitMQ.Client.ConnectionFactory	Main entry point to the RabbitMQ .NET AMQP client API. Constructs IConnection instances.
RabbitMQ.Client.ConnectionFactory.ObtainSocket	(undocumented)
RabbitMQ.Client.Content.BasicMessageBuilder	Framework for constructing various types of AMQP Basic-class applicatio messages.
RabbitMQ.Client.Content.BasicMessageReader	Framework for analyzing various types of AMQP Basic-class applicatio messages.
RabbitMQ.Client.Content.BytesMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "BytesMessage" wire encodin
RabbitMQ.Client.Content.BytesMessageReader	Analyzes AMQP Basic-class message binary-compatible with QPid's "BytesMessage" wire encoding.
RabbitMQ.Client.Content.BytesWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "BytesMessage" wire encoding.

RabbitMQ.Client.Content.IBytesMessageBuilder	Interface for constructing messages binary-compatible with QPid's "BytesMessage" wire encoding.
RabbitMQ.Client.Content.IBytesMessageReader	Analyzes messages binary-compatibl with QPid's "BytesMessage" wire encoding.
RabbitMQ.Client.Content.IMapMessageBuilder	Interface for constructing messages binary-compatible with QPid's "MapMessage" wire encoding.
RabbitMQ.Client.Content.IMapMessageReader	Analyzes messages binary-compatibl with QPid's "MapMessage" wire encoding.
RabbitMQ.Client.Content.IMessageBuilder	Interface for constructing applicatio messages.
RabbitMQ.Client.Content.IMessageReader	Interface for analyzing application messages.
RabbitMQ.Client.Content.IStreamMessageBuilder	Interface for constructing messages binary-compatible with QPid's "StreamMessage" wire encoding.
RabbitMQ.Client.Content.IStreamMessageReader	Analyzes messages binary-compatibl with QPid's "StreamMessage" wire encoding.
RabbitMQ.Client.Content.MapMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "MapMessage" wire encoding
RabbitMQ.Client.Content.MapMessageReader	Analyzes AMQP Basic-class message binary-compatible with QPid's "MapMessage" wire encoding.
RabbitMQ.Client.Content.MapWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "MapMessage" wire encoding.
RabbitMQ.Client.Content.PrimitiveParser	Utility class for extracting typed values from strings.
RabbitMQ.Client.Content.StreamMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "StreamMessage" wire encoding.
RabbitMQ.Client.Content.StreamMessageReader	Analyzes AMQP Basic-class message binary-compatible with QPid's "StreamMessage" wire encoding.
RabbitMQ.Client.Content.StreamWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "StreamMessage" wire encoding.
RabbitMQ.Client.Content.StreamWireFormattingTag	Tags used in parsing and generating StreamWireFormatting message bodies.
RabbitMQ.Client.DefaultBasicConsumer	Useful default/base implementation IBasicConsumer. Subclass and override HandleBasicDeliver in application code.
RabbitMQ.Client.Events.BasicAckEventArgs	Contains all the information about a message acknowledged from an AMQP broker within the Basic content-class.
RabbitMQ.Client.Events.BasicAckEventHandler	

	Delegate used to process Basic.Ack events.
RabbitMQ.Client.Events.BasicDeliverEventArgs	Contains all the information about a message delivered from an AMQP broker within the Basic content-clas
RabbitMQ.Client.Events.BasicDeliverEventHandler	Delegate used to process Basic.Deliver events.
RabbitMQ.Client.Events.BasicNackEventArgs	Contains all the information about a message nack'd from an AMQP brok within the Basic content-class.
RabbitMQ.Client.Events.BasicNackEventHandler	Delegate used to process Basic.Nack events.
RabbitMQ.Client.Events.BasicRecoverOkEventHandler	Delegate used to process Basic.RecoverOk events.
RabbitMQ.Client.Events.BasicReturnEventArgs	Contains all the information about a message returned from an AMQP broker within the Basic content-clas
RabbitMQ.Client.Events.BasicReturnEventHandler	Delegate used to process Basic.Return events.
RabbitMQ.Client.Events.CallbackExceptionEventArgs	Describes an exception that was thrown during the library's invocatio of an application-supplied callback handler.
RabbitMQ.Client.Events.CallbackExceptionEventHandler	Callback invoked when other callbacks throw unexpected exceptions.
RabbitMQ.Client.Events.ConnectionBlockedEventArgs	Event relating to connection being blocked
RabbitMQ.Client.Events.ConnectionBlockedEventHandler	Delegate used to process connection blocked events.
RabbitMQ.Client.Events.ConnectionShutdownEventHandler	Delegate used to process connection shutdown notifications.
RabbitMQ.Client.Events.ConnectionUnblockedEventHandler	Delegate used to process connection unblocked events.
RabbitMQ.Client.Events.ConsumerCancelledEventHandler	Delegate used to process consumer cancellations.
RabbitMQ.Client.Events.ConsumerEventArgs	Event relating to a successful consumer registration or cancellatio
RabbitMQ.Client.Events.ConsumerEventHandler	Callback for events relating to consumer registration and cancellation.
RabbitMQ.Client.Events.ConsumerShutdownEventHandler	Callback for events relating to consumer shutdown.
RabbitMQ.Client.Events.EventingBasicConsumer	Experimental class exposing an IBasicConsumer's methods as separate events.
RabbitMQ.Client.Events.FlowControlEventArgs	Event relating to flow control
RabbitMQ.Client.Events.FlowControlEventHandler	Delegate used to process flow contr events.
RabbitMQ.Client.Events.ModelShutdownEventHandler	Delegate used to process model shutdown notifications.
RabbitMQ.Client.Exceptions.AlreadyClosedException	Thrown when the application tries t make use of a session or connection that has already been shut down.
RabbitMQ.Client.Exceptions.AuthenticationFailureException	Thrown when the cause is an authentication failure.

RabbitMQ.Client.Exceptions.BrokerUnreachableException	Thrown when no connection could b opened during a ConnectionFactory.CreateConnectio attempt.
RabbitMQ.Client.Exceptions.ChannelAllocationException	Thrown when a SessionManager cannot allocate a new channel number, or the requested channel number is already in use.
RabbitMQ.Client.Exceptions.ConnectFailureException	Thrown when a connection to the broker fails
RabbitMQ.Client.Exceptions.OperationInterruptedException	Thrown when a session is destroyed during an RPC call to a broker. For example, if a TCP connection dropping causes the destruction of a session in the middle of a QueueDeclare operation, an OperationInterruptedException will be thrown to the caller of IModel.QueueDeclare.
RabbitMQ.Client.Exceptions.PacketNotRecognizedException	Thrown to indicate that the peer didn't understand the packet receive from the client. Peer sent default message describing protocol version it is using and transport parameters
RabbitMQ.Client.Exceptions.PossibleAuthenticationFailureException	Thrown when the likely cause is an authentication failure.
RabbitMQ.Client.Exceptions.ProtocolVersionMismatchException	Thrown to indicate that the peer doe not support the wire protocol versio we requested immediately after opening the TCP socket.
RabbitMQ.Client.Exceptions.UnexpectedMethodException	Thrown when the model receives an RPC reply that it wasn't expecting.
RabbitMQ.Client.Exceptions.UnsupportedMethodException	Thrown when the model receives an RPC request it cannot satisfy.
RabbitMQ.Client.Exceptions.UnsupportedMethodFieldException	Thrown when the model cannot transmit a method field because the version of the protocol the model is implementing does not contain a definition for the field in question.
RabbitMQ.Client.Exceptions.WireFormattingException	Thrown when the wire-formatting code cannot encode a particular .NE value to AMQP protocol format.
RabbitMQ.Client.ExchangeType	Convenience class providing compile-time names for standard exchange types.
RabbitMQ.Client.ExternalMechanism	(undocumented)
RabbitMQ.Client.ExternalMechanismFactory	(undocumented)
RabbitMQ.Client.IBasicConsumer	Consumer interface for Basic content-class. Used to receive messages from a queue by subscription.
RabbitMQ.Client.IBasicProperties	Common AMQP Basic content-class headers interface, spanning the unio of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 AMQP.
RabbitMQ.Client.IConnection	Main interface to an AMQP connection.

RabbitMQ.Client.IContentHeader	A decoded AMQP content header frame.
RabbitMQ.Client.IFileProperties	Common AMQP File content-class headers interface, spanning the unio of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 AMQP.
RabbitMQ.Client.IMethod	A decoded AMQP method frame.
RabbitMQ.Client.IModel	Common AMQP model, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 AMQP.
RabbitMQ.Client.IProtocol	Object describing various overarchi parameters associated with a particular AMQP protocol variant.
RabbitMQ.Client.IStreamProperties	Common AMQP Stream content-clas headers interface, spanning the unio of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 AMQP.
RabbitMQ.Client.MessagePatterns.SimpleRpcClient	Implements a simple RPC client.
RabbitMQ.Client.MessagePatterns.SimpleRpcServer	Implements a simple RPC service, responding to requests received via Subscription.
RabbitMQ.Client.MessagePatterns.Subscription	Manages a subscription to a queue exchange.
RabbitMQ.Client.PlainMechanism	(undocumented)
RabbitMQ.Client.PlainMechanismFactory	(undocumented)
RabbitMQ.Client.Protocols	Concrete, predefined IProtocol instances ready for use with ConnectionFactory.
RabbitMQ.Client.PublicationAddress	Container for an exchange name, exchange type and routing key, usab as the target address of a message t be published.
RabbitMQ.Client.QueueDeclareOk	(undocumented)
RabbitMQ.Client.QueueingBasicConsumer	Simple IBasicConsumer implementation that uses a SharedQueue to buffer incoming deliveries.
RabbitMQ.Client.ShutdownEventArgs	Information about the reason why a particular model, session, or connection was destroyed.
RabbitMQ.Client.ShutdownInitiator	Describes the source of a shutdown event.
RabbitMQ.Client.ShutdownReportEntry	Single entry object in the shutdown report that encapsulates description of the error which occured during shutdown
RabbitMQ.Client.SslHelper	Represents an SslHelper which does the actual heavy lifting to set up an SSL connection, using the config options in an SslOption to make things cleaner
RabbitMQ.Client.SslOption	Represents a configurable SSL optio used in setting up an SSL connectio
RabbitMQ.Util.BlockingCell	

	A thread-safe single-assignment reference cell.
RabbitMQ.Util.DebugUtil	Miscellaneous debugging and development utilities.
RabbitMQ.Util.Either	Models the disjoint union of two alternatives, a "left" alternative and "right" alternative.
RabbitMQ.Util.EitherAlternative	Used internally by class Either.
RabbitMQ.Util.IntAllocator	(undocumented)
RabbitMQ.Util.IntAllocator.IntervalList	(undocumented)
RabbitMQ.Util.NetworkBinaryReader	Subclass of BinaryReader that reads integers etc in correct network orde
RabbitMQ.Util.NetworkBinaryWriter	Subclass of BinaryWriter that writes integers etc in correct network orde
RabbitMQ.Util.SharedQueue	A thread-safe shared queue implementation.
RabbitMQ.Util.SharedQueue<T>	A thread-safe shared queue implementation.
RabbitMQ.Util.SharedQueueEnumerator<T>	Implementation of the IEnumerator interface, for permitting SharedQue to be used in foreach loops.
RabbitMQ.Util.SynchronizedList<T>	(undocumented)
RabbitMQ.Util.XmlUtil	Miscellaneous helpful XML utilities.

Index

Namespace RabbitMQ.Client

Summary

Main public API to the RabbitMQ .NET AMQP client library.

Types

Type	Summary
AmqpTcpEndpoint	Represents a TCP-addressable AMQP peer, including the protocol variant to use, and a host name and port number.
AmqpTimestamp	Structure holding an AMQP timestamp, a posix 64-bit time_t.
AmqpVersion	Represents a version of the AMQP specification.
AuthMechanism	A pluggable authentication mechanism.
AuthMechanismFactory	(undocumented)
BasicGetResult	Represents Basic.GetOk responses from the server.
BinaryTableValue	Wrapper for a byte[]. May appear as values read from and written to AMQP field tables.
ConnectionFactory	Main entry point to the RabbitMQ .NET AMQP client API. Constructs IConnection instances.
DefaultBasicConsumer	Useful default/base implementation of IBasicConsumer. Subclass and override HandleBasicDeliver in application code.
ExchangeType	Convenience class providing compile-time names for standard exchange types.
ExternalMechanism	(undocumented)
ExternalMechanismFactory	(undocumented)
IBasicConsumer	Consumer interface for Basic content-class. Used to receive messages from a queue by subscription.
IBasicProperties	Common AMQP Basic content-class headers interface, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 of AMQP.
IConnection	Main interface to an AMQP connection.
IContentHeader	A decoded AMQP content header frame.
IFileProperties	Common AMQP File content-class headers interface, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 of AMQP.
IMethod	A decoded AMQP method frame.
IModel	Common AMQP model, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 of AMQP.
IProtocol	Object describing various overarching parameters associated with a particular AMQP protocol variant.
IStreamProperties	Common AMQP Stream content-class headers interface, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 of AMQP.
ConnectionFactory.ObtainSocket	(undocumented)
PlainMechanism	(undocumented)
PlainMechanismFactory	(undocumented)
Protocols	Concrete, predefined IProtocol instances ready for use with ConnectionFactory.
PublicationAddress	Container for an exchange name, exchange type and routing key, usable as the target address of a message to be published.
QueueDeclareOk	(undocumented)
QueueingBasicConsumer	Simple IBasicConsumer implementation that uses a SharedQueue to buffer incoming deliveries.

ShutdownEventArgs	Information about the reason why a particular model, session, or connection was destroyed.
ShutdownInitiator	Describes the source of a shutdown event.
ShutdownReportEntry	Single entry object in the shutdown report that encapsulates description of the error which occured during shutdown
SslHelper	Represents an SslHelper which does the actual heavy lifting to set up an SSL connection, using the config options in an SslOption to make things cleaner
SslOption	Represents a configurable SSL option, used in setting up an SSL connection.

Index | Namespace RabbitMQ.Client

public class AmqpTcpEndpoint

Summary

Represents a TCP-addressable AMQP peer, including the protocol variant to use, and a host name and port number.

Para

Some of the constructors take, as a convenience, a System.Uri instance representing an AMQP server address. The use of Uri here is not standardised - Uri is simply a convenient container for internet-address-like components. In particular, the Uri "Scheme" property is ignored: only the "Host" and "Port" properties are extracted.

Field Summary

Flags	Type	Name	Summary
public const	`int`	`DefaultAmqpSslPort`	Indicates that the default port for the protocol should be used
public const	`int`	`UseDefaultPort`	(undocumented)

Property Summary

Flags	Type	Name	Summary
public	`string`	`HostName` (rw)	Retrieve or set the hostname of this AmqpTcpEndpoint.
public	`int`	`Port` (rw)	Retrieve or set the port number of this AmqpTcpEndpoint. A port number of -1 causes the default port number for the IProtocol to be used.
public	`IProtocol`	`Protocol` (rw)	Retrieve or set the IProtocol of this AmqpTcpEndpoint.
public	`SslOption`	`Ssl` (rw)	Retrieve the SSL options for this AmqpTcpEndpoint. If not set, null is returned

Constructor Summary

Flags	Name	Summary
public	`AmqpTcpEndpoint()`	Construct an AmqpTcpEndpoint with a hostname of "localhost", using the IProtocol from Protocols.FromEnvironment(), and the default port number of that IProtocol.
public	`AmqpTcpEndpoint(string hostName)`	Construct an AmqpTcpEndpoint with the given hostname, using the IProtocol from Protocols.FromEnvironment(), and the default port number of that IProtocol.
public	`AmqpTcpEndpoint(IProtocol protocol, Uri uri, SslOption ssl)`	Construct an AmqpTcpEndpoint with the given IProtocol, Uri and ssl options.
public	`AmqpTcpEndpoint(Uri uri)`	Construct an AmqpTcpEndpoint with the given Uri, using the IProtocol from Protocols.FromEnvironment().
public	`AmqpTcpEndpoint(IProtocol protocol, Uri uri)`	Construct an AmqpTcpEndpoint with the given IProtocol and Uri.
public	`AmqpTcpEndpoint(IProtocol protocol, string hostName, int portOrMinusOne)`	Construct an AmqpTcpEndpoint with the given IProtocol, hostname, and port number. If the port number is -1, the default port number for the IProtocol will be used.
public	`AmqpTcpEndpoint(IProtocol protocol, string hostName, int portOrMinusOne, SslOption ssl)`	Construct an AmqpTcpEndpoint with the given IProtocol, hostname, port number and ssl option. If the port number is -1, the default port number for the IProtocol will be used.
public	`AmqpTcpEndpoint(IProtocol protocol, string hostName)`	Construct an AmqpTcpEndpoint with the given IProtocol and hostname, using the default port for the IProtocol.
public		

	AmqpTcpEndpoint(string hostName, int portOrMinusOne)	Construct an AmqpTcpEndpoint with the given hostname and port number, using the IProtocol from Protocols.FromEnvironment(). If the port number is -1, the default port number for the IProtocol will be used.
public	AmqpTcpEndpoint(IProtocol protocol)	Construct an AmqpTcpEndpoint with the given IProtocol, "localhost" as the hostname, and using the default port for the IProtocol.

Method Summary

Flags	Name	Summary
public virtual	bool Equals(object obj)	Compares this instance by value (protocol, hostname, port) against another instance
public virtual	int GetHashCode()	Implementation of hash code depending on protocol, hostname and port, to line up with the implementation of Equals()
public static	AmqpTcpEndpoint Parse(IProtocol protocol, string address)	Construct an instance from a protocol and an address in "hostname:port" format.
public static	AmqpTcpEndpoint[] ParseMultiple(IProtocol protocol, string addresses)	Splits the passed-in string on ",", and passes the substrings to AmqpTcpEndpoint.Parse()
public virtual	string ToString()	Returns a URI-like string of the form amqp-PROTOCOL://HOSTNAME:PORTNUMBER

Field Detail

public const int DefaultAmqpSslPort

Summary

Indicates that the default port for the protocol should be used

public const int UseDefaultPort

Property Detail

public string HostName (rw)

Summary

Retrieve or set the hostname of this AmqpTcpEndpoint.

public int Port (rw)

Summary

Retrieve or set the port number of this AmqpTcpEndpoint. A port number of -1 causes the default port number for the IProtocol to be used.

public IProtocol Protocol (rw)

Summary

Retrieve or set the IProtocol of this AmqpTcpEndpoint.

public SslOption Ssl (rw)

Summary

Retrieve the SSL options for this AmqpTcpEndpoint. If not set, null is returned

Constructor Detail

AmqpTcpEndpoint

```
public AmqpTcpEndpoint()
```

Summary

Construct an AmqpTcpEndpoint with a hostname of "localhost", using the IProtocol from Protocols.FromEnvironment(), and the default port number of that IProtocol.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(string hostName)
```

	Name	Type
Parameters	hostName	`string`

Summary

Construct an AmqpTcpEndpoint with the given hostname, using the IProtocol from Protocols.FromEnvironment(), and the default port number of that IProtocol.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol, Uri uri, SslOption ssl)
```

	Name	Type
Parameters	protocol	`IProtocol`
	uri	`Uri`
	ssl	`SslOption`

Summary

Construct an AmqpTcpEndpoint with the given IProtocol, Uri and ssl options.

Remarks

Please see the class overview documentation for information about the Uri format in use.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(Uri uri)
```

	Name	Type
Parameters	uri	`Uri`

Summary

Construct an AmqpTcpEndpoint with the given Uri, using the IProtocol from Protocols.FromEnvironment().

Remarks

Please see the class overview documentation for information about the Uri format in use.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol, Uri uri)
```

	Name	Type
Parameters	protocol	IProtocol
	uri	Uri

Summary

Construct an AmqpTcpEndpoint with the given IProtocol and Uri.

Remarks

Please see the class overview documentation for information about the Uri format in use.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol, string hostName, int portOrMinusOne)
```

	Name	Type
Parameters	protocol	IProtocol
	hostName	string
	portOrMinusOne	int

Summary

Construct an AmqpTcpEndpoint with the given IProtocol, hostname, and port number. If the port number is -1, the default port number for the IProtocol will be used.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol, string hostName, int portOrMinusOne, SslOption
ssl)
```

	Name	Type
	protocol	IProtocol
Parameters	hostName	string
	portOrMinusOne	int
	ssl	SslOption

Summary

Construct an AmqpTcpEndpoint with the given IProtocol, hostname, port number and ssl option. If the port number is -1, the default port number for the IProtocol will be used.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol, string hostName)
```

	Name	Type
Parameters	protocol	IProtocol
	hostName	string

Summary

Construct an AmqpTcpEndpoint with the given IProtocol and hostname, using the default port for the IProtocol.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(string hostName, int portOrMinusOne)
```

	Name	Type
Parameters	hostName	`string`
	portOrMinusOne	`int`

Summary

Construct an AmqpTcpEndpoint with the given hostname and port number, using the IProtocol from Protocols.FromEnvironment(). If the port number is -1, the default port number for the IProtocol will be used.

AmqpTcpEndpoint

```
public AmqpTcpEndpoint(IProtocol protocol)
```

	Name	Type
Parameters	protocol	`IProtocol`

Summary

Construct an AmqpTcpEndpoint with the given IProtocol, "localhost" as the hostname, and using the default port for the IProtocol.

Method Detail

Equals

```
public virtual bool Equals(object obj)
```

Flags	public virtual	
Return type	`bool`	
Parameters	**Name**	**Type**
	obj	`object`

Summary

Compares this instance by value (protocol, hostname, port) against another instance

GetHashCode

```
public virtual int GetHashCode()
```

Flags	public virtual
Return type	`int`

Summary

Implementation of hash code depending on protocol, hostname and port, to line up with the implementation of Equals()

Parse

```
public static AmqpTcpEndpoint Parse(IProtocol protocol, string address)
```

Flags	public static
Return type	`AmqpTcpEndpoint`

	Name	Type
Parameters	protocol	IProtocol
	address	string

Summary

Construct an instance from a protocol and an address in "hostname:port" format.

Remarks

If the address string passed in contains ":", it is split into a hostname and a port-number part. Otherwise, the entire string is used as the hostname, and the port-number is set to -1 (meaning the default number for the protocol variant specified). Hostnames provided as IPv6 must appear in square brackets ([]).

ParseMultiple

```
public static AmqpTcpEndpoint[] ParseMultiple(IProtocol protocol, string addresses)
```

	Name	Type
Flags	public static	
Return type	AmqpTcpEndpoint[]	
Parameters	protocol	IProtocol
	addresses	string

Summary

Splits the passed-in string on ",", and passes the substrings to AmqpTcpEndpoint.Parse()

Remarks

Accepts a string of the form "hostname:port, hostname:port, ...", where the ":port" pieces are optional, and returns a corresponding array of AmqpTcpEndpoints.

ToString

```
public virtual string ToString()
```

Flags	public virtual
Return type	string

Summary

Returns a URI-like string of the form amqp-PROTOCOL://HOSTNAME:PORTNUMBER

Remarks

This method is intended mainly for debugging and logging use.

Index

Namespace RabbitMQ.Client.Content

Summary

Public API for construction and analysis of messages that are binary-compatible with messages produced and consumed by QPid's JMS compatibility layer.

Types

Type	Summary
BasicMessageBuilder	Framework for constructing various types of AMQP Basic-class application messages.
BasicMessageReader	Framework for analyzing various types of AMQP Basic-class application messages.
BytesMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "BytesMessage" wire encoding.
BytesMessageReader	Analyzes AMQP Basic-class messages binary-compatible with QPid's "BytesMessage" wire encoding.
BytesWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "BytesMessage" wire encoding.
IBytesMessageBuilder	Interface for constructing messages binary-compatible with QPid's "BytesMessage" wire encoding.
IBytesMessageReader	Analyzes messages binary-compatible with QPid's "BytesMessage" wire encoding.
IMapMessageBuilder	Interface for constructing messages binary-compatible with QPid's "MapMessage" wire encoding.
IMapMessageReader	Analyzes messages binary-compatible with QPid's "MapMessage" wire encoding.
IMessageBuilder	Interface for constructing application messages.
IMessageReader	Interface for analyzing application messages.
IStreamMessageBuilder	Interface for constructing messages binary-compatible with QPid's "StreamMessage" wire encoding.
IStreamMessageReader	Analyzes messages binary-compatible with QPid's "StreamMessage" wire encoding.
MapMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "MapMessage" wire encoding.
MapMessageReader	Analyzes AMQP Basic-class messages binary-compatible with QPid's "MapMessage" wire encoding.
MapWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "MapMessage" wire encoding.
PrimitiveParser	Utility class for extracting typed values from strings.
StreamMessageBuilder	Constructs AMQP Basic-class messages binary-compatible with QPid's "StreamMessage" wire encoding.
StreamMessageReader	Analyzes AMQP Basic-class messages binary-compatible with QPid's "StreamMessage" wire encoding.
StreamWireFormatting	Internal support class for use in reading and writing information binary-compatible with QPid's "StreamMessage" wire encoding.
StreamWireFormattingTag	Tags used in parsing and generating StreamWireFormatting message bodies.

Index | Namespace RabbitMQ.Client.Content

public class BasicMessageBuilder

- implements IMessageBuilder

Summary

Framework for constructing various types of AMQP Basic-class application messages.

Field Summary

Flags	Type	Name	Summary
public const	int	DefaultAccumulatorSize	By default, new instances of BasicMessageBuilder and its subclasses will have this much initial buffer space.

Property Summary

Flags	Type	Name	Summary
public virtual final	Stream	BodyStream (r)	Implement IMessageBuilder.BodyStream
public virtual final	IDictionary<string,object>	Headers (r)	Implement IMessageBuilder.Headers
public	IBasicProperties	Properties (r)	Retrieve the IBasicProperties associated with this instance.
public	NetworkBinaryWriter	Writer (r)	Retrieve this instance's NetworkBinaryWriter writing to BodyStream.

Constructor Summary

Flags	Name	Summary
public	BasicMessageBuilder(IModel model, int initialAccumulatorSize)	Construct an instance ready for writing.
public	BasicMessageBuilder(IModel model)	Construct an instance ready for writing.

Method Summary

Flags	Name	Summary
public virtual	byte[] GetContentBody()	Implement IMessageBuilder.GetContentBody
public virtual	IContentHeader GetContentHeader()	Implement IMessageBuilder.GetContentHeader
public virtual	string GetDefaultContentType()	Implement IMessageBuilder.GetDefaultContentType(). Returns null; overridden in subclasses.
public virtual final	IMessageBuilder RawWrite(byte b)	Implement IMessageBuilder.RawWrite
public virtual final	IMessageBuilder RawWrite(byte[] bytes)	Implement IMessageBuilder.RawWrite
public virtual final	IMessageBuilder RawWrite(byte[] bytes, int offset, int length)	Implement IMessageBuilder.RawWrite

Field Detail

public const int DefaultAccumulatorSize

Summary

By default, new instances of BasicMessageBuilder and its subclasses will have this much initial buffer space.

Property Detail

public virtual final Stream BodyStream (r)

Summary

Implement IMessageBuilder.BodyStream

public virtual final IDictionary<string,object> Headers (r)

Summary

Implement IMessageBuilder.Headers

public IBasicProperties Properties (r)

Summary

Retrieve the IBasicProperties associated with this instance.

public NetworkBinaryWriter Writer (r)

Summary

Retrieve this instance's NetworkBinaryWriter writing to BodyStream.

Remarks

If no NetworkBinaryWriter instance exists, one is created, pointing at the beginning of the accumulator. If one already exists, the existing instance is returned. The instance is not reset.

Constructor Detail

BasicMessageBuilder

```
public BasicMessageBuilder(IModel model, int initialAccumulatorSize)
```

	Name	Type
Parameters	model	IModel
	initialAccumulatorSize	int

Summary

Construct an instance ready for writing.

BasicMessageBuilder

```
public BasicMessageBuilder(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Construct an instance ready for writing.

Remarks

The DefaultAccumulatorSize is used for the initial accumulator buffer size.

Method Detail

GetContentBody

```
public virtual byte[] GetContentBody()
```

Flags public virtual
Return type `byte[]`
Summary

Implement IMessageBuilder.GetContentBody

GetContentHeader

```
public virtual IContentHeader GetContentHeader()
```

Flags public virtual
Return type IContentHeader
Summary

Implement IMessageBuilder.GetContentHeader

GetDefaultContentType

```
public virtual string GetDefaultContentType()
```

Flags public virtual
Return type `string`
Summary

Implement IMessageBuilder.GetDefaultContentType(). Returns null; overridden in subclasses.

RawWrite

```
public virtual final IMessageBuilder RawWrite(byte b)
```

Flags public virtual final
Return type IMessageBuilder

Parameters

Name	Type
b	byte

Summary

Implement IMessageBuilder.RawWrite

RawWrite

```
public virtual final IMessageBuilder RawWrite(byte[] bytes)
```

Flags public virtual final
Return type IMessageBuilder

Parameters

Name	Type
bytes	byte[]

Summary

Implement IMessageBuilder.RawWrite

RawWrite

```
public virtual final IMessageBuilder RawWrite(byte[] bytes, int offset, int length)
```

Flags	public virtual final
Return type	IMessageBuilder

Parameters

Name	Type
bytes	byte[]
offset	int
length	int

Summary

Implement IMessageBuilder.RawWrite

Index

Namespace RabbitMQ.Client.Events

Summary

Public API for various events and event handlers that are part of the AMQP client library.

Types

Type	Summary
BasicAckEventArgs	Contains all the information about a message acknowledged from an AMQP broker within the Basic content-class.
BasicAckEventHandler	Delegate used to process Basic.Ack events.
BasicDeliverEventArgs	Contains all the information about a message delivered from an AMQP broker within the Basic content-class.
BasicDeliverEventHandler	Delegate used to process Basic.Deliver events.
BasicNackEventArgs	Contains all the information about a message nack'd from an AMQP broker within the Basic content-class.
BasicNackEventHandler	Delegate used to process Basic.Nack events.
BasicRecoverOkEventHandler	Delegate used to process Basic.RecoverOk events.
BasicReturnEventArgs	Contains all the information about a message returned from an AMQP broker within the Basic content-class.
BasicReturnEventHandler	Delegate used to process Basic.Return events.
CallbackExceptionEventArgs	Describes an exception that was thrown during the library's invocation of an application-supplied callback handler.
CallbackExceptionEventHandler	Callback invoked when other callbacks throw unexpected exceptions.
ConnectionBlockedEventArgs	Event relating to connection being blocked
ConnectionBlockedEventHandler	Delegate used to process connection blocked events.
ConnectionShutdownEventHandler	Delegate used to process connection shutdown notifications.
ConnectionUnblockedEventHandler	Delegate used to process connection unblocked events.
ConsumerCancelledEventHandler	Delegate used to process consumer cancellations.
ConsumerEventArgs	Event relating to a successful consumer registration or cancellation.
ConsumerEventHandler	Callback for events relating to consumer registration and cancellation.
ConsumerShutdownEventHandler	Callback for events relating to consumer shutdown.
EventingBasicConsumer	Experimental class exposing an IBasicConsumer's methods as separate events.
FlowControlEventArgs	Event relating to flow control
FlowControlEventHandler	Delegate used to process flow control events.
ModelShutdownEventHandler	Delegate used to process model shutdown notifications.

Index | Namespace RabbitMQ.Client.Events

public class BasicAckEventArgs

- extends `EventArgs`

Summary

Contains all the information about a message acknowledged from an AMQP broker within the Basic content-class.

Property Summary

Flags	Type	Name	Summary
public	`ulong`	`DeliveryTag` (rw)	The sequence number of the acknowledged message, or the closed upper bound of acknowledged messages if multiple is true.
public	`bool`	`Multiple` (rw)	Whether this acknoledgement applies to one message or multiple messages.

Constructor Summary

Flags	Name	Summary
public	`BasicAckEventArgs()`	Default constructor.

Property Detail

public ulong DeliveryTag (rw)

Summary

The sequence number of the acknowledged message, or the closed upper bound of acknowledged messages if multiple is true.

public bool Multiple (rw)

Summary

Whether this acknoledgement applies to one message or multiple messages.

Constructor Detail

BasicAckEventArgs

`public BasicAckEventArgs()`
Summary

Default constructor.
Index

Namespace RabbitMQ.Client.Exceptions

Summary

Public API for exceptions visible to the user of the AMQP client library.

Types

Type	Summary
AlreadyClosedException	Thrown when the application tries to make use of a session or connection that has already been shut down.
AuthenticationFailureException	Thrown when the cause is an authentication failure.
BrokerUnreachableException	Thrown when no connection could be opened during a ConnectionFactory.CreateConnection attempt.
ChannelAllocationException	Thrown when a SessionManager cannot allocate a new channel number, or the requested channel number is already in use.
ConnectFailureException	Thrown when a connection to the broker fails
OperationInterruptedException	Thrown when a session is destroyed during an RPC call to a broker. For example, if a TCP connection dropping causes the destruction of a session in the middle of a QueueDeclare operation, an OperationInterruptedException will be thrown to the caller of IModel.QueueDeclare.
PacketNotRecognizedException	Thrown to indicate that the peer didn't understand the packet received from the client. Peer sent default message describing protocol version it is using and transport parameters.
PossibleAuthenticationFailureException	Thrown when the likely cause is an authentication failure.
ProtocolVersionMismatchException	Thrown to indicate that the peer does not support the wire protocol version we requested immediately after opening the TCP socket.
UnexpectedMethodException	Thrown when the model receives an RPC reply that it wasn't expecting.
UnsupportedMethodException	Thrown when the model receives an RPC request it cannot satisfy.
UnsupportedMethodFieldException	Thrown when the model cannot transmit a method field because the version of the protocol the model is implementing does not contain a definition for the field in question.
WireFormattingException	Thrown when the wire-formatting code cannot encode a particular .NET value to AMQP protocol format.

Index | Namespace RabbitMQ.Client.Exceptions

public class AlreadyClosedException

- extends OperationInterruptedException

Summary

Thrown when the application tries to make use of a session or connection that has already been shut down.

Constructor Summary

Flags	Name	Summary
public	AlreadyClosedException(ShutdownEventArgs reason)	Construct an instance containing the given shutdown reason.

Constructor Detail

AlreadyClosedException

```
public AlreadyClosedException(ShutdownEventArgs reason)
```

Parameters	Name	Type
	reason	ShutdownEventArgs

Summary

Construct an instance containing the given shutdown reason.

Index

Namespace RabbitMQ.Client.MessagePatterns

Summary

Public API for high-level helper classes and interface for common ways of using the AMQP client library.

Types

Type	Summary
SimpleRpcClient	Implements a simple RPC client.
SimpleRpcServer	Implements a simple RPC service, responding to requests received via a Subscription.
Subscription	Manages a subscription to a queue or exchange.

Index | Namespace RabbitMQ.Client.MessagePatterns

public class SimpleRpcClient

- implements `IDisposable`

Summary

Implements a simple RPC client.

Remarks

This class sends requests that can be processed by remote SimpleRpcServer instances.

The basic pattern for accessing a remote service is to determine the exchange name and routing key needed for submissions of service requests, and to construct a SimpleRpcClient instance using that address. Once constructed, the various Call() and Cast() overloads can be used to send requests and receive the corresponding replies.

```
string queueName = "ServiceRequestQueue"; // See also Subscription ctors
using (IConnection conn = new ConnectionFactory()
                                .CreateConnection(serverAddress)) {
    using (IModel ch = conn.CreateModel()) {
        SimpleRpcClient client =
            new SimpleRpcClient(ch, queueName);
        client.TimeoutMilliseconds = 5000; // optional

        /// ... make use of the various Call() overloads
    }
}
```

Instances of this class declare a queue, so it is the user's responsibility to ensure that the exchange concerned exists (using IModel.ExchangeDeclare) before invoking Call() or Cast().

This class implements only a few basic RPC message formats - to extend it with support for more formats, either subclass, or transcode the messages before transmission using the built-in byte[] format.

See

- RabbitMQ.Client.MessagePatterns.SimpleRpcServer

Property Summary

Flags	Type	Name	Summary
public	PublicationAddress	Address (rw)	Retrieve or modify the address that will be used for the next Call() or Cast().
public	IModel	Model (r)	Retrieve the IModel this instance uses to communicate.
public	Subscription	Subscription (r)	Retrieve the Subscription that is used to receive RPC replies corresponding to Call() RPC requests. May be null.
public	int	TimeoutMilliseconds (rw)	Retrieve or modify the timeout (in milliseconds) that will be used for the next Call().

Event Summary

Type	Name	Summary
EventHandler	Disconnected	This event is fired whenever Call() detects the disconnection of the underlying Subscription while waiting for a reply from the service.
EventHandler	TimedOut	This event is fired whenever Call() decides that a timeout has occurred while waiting for a reply from the service.

Constructor Summary

Flags	Name	Summary
public	SimpleRpcClient(IModel model, string exchange, string exchangeType, string routingKey)	Construct an instance that will deliver to the named and typed exchange, with the given routing key.
public	SimpleRpcClient(IModel model, PublicationAddress address)	Construct an instance that will deliver to the given address.
public	SimpleRpcClient(IModel model)	Construct an instance with no configured Address. The Address property must be set before Call() or Cast() are called.
public	SimpleRpcClient(IModel model, string queueName)	Construct an instance that will deliver to the default exchange (""), with routing key equal to the passed in queueName, thereby delivering directly to a named queue on the AMQP server.

Method Summary

Flags	Name	Summary
public virtual	byte[] Call(byte[] body)	Sends a simple byte[] message, without any custom headers or properties.
public virtual	byte[] Call(IBasicProperties requestProperties, byte[] body, out IBasicProperties replyProperties)	Sends a byte[] message and IBasicProperties header, returning both the body and headers of the received reply.
public virtual	object[] Call(object[] args)	Sends a "jms/stream-message"-encoded RPC request, and expects an RPC reply in the same format.
public virtual	BasicDeliverEventArgs Call(IBasicProperties requestProperties, byte[] body)	Sends a byte[]/IBasicProperties RPC request, returning full information about the delivered reply as a BasicDeliverEventArgs.
public virtual	void Cast(IBasicProperties requestProperties, byte[] body)	Sends an asynchronous/one-way message to the service.
public	void Close()	Close the reply subscription associated with this instance, if any.
public virtual	void OnDisconnected()	Signals that the Subscription we use for receiving our RPC replies was disconnected while we were waiting.
public virtual	void OnTimedOut()	Signals that the configured timeout fired while waiting for an RPC reply.

Property Detail

public PublicationAddress Address (rw)

Summary

Retrieve or modify the address that will be used for the next Call() or Cast().

Remarks

This address represents the service, i.e. the destination service requests should be published to. It can be changed at any time before a Call() or Cast() request is sent - the value at the time of the call is used by Call() and Cast().

public IModel Model (r)

Summary

Retrieve the IModel this instance uses to communicate.

public Subscription Subscription (r)

Summary

Retrieve the Subscription that is used to receive RPC replies corresponding to Call() RPC requests. May be null.
Remarks

Upon construction, this property will be null. It is initialised by the protected virtual method EnsureSubscription upon the first call to Call(). Calls to Cast() do not initialise the subscription, since no replies are expected or possible when using Cast().

public int TimeoutMilliseconds (rw)

Summary

Retrieve or modify the timeout (in milliseconds) that will be used for the next Call().
Remarks

This property defaults to System.Threading.Timeout.Infinite (i.e. -1). If it is set to any other value, Call() will only wait for the specified amount of time before returning indicating a timeout.

See also TimedOut event and OnTimedOut().

Event Detail

EventHandler Disconnected

Summary

This event is fired whenever Call() detects the disconnection of the underlying Subscription while waiting for a reply from the service.
Remarks

See also OnDisconnected(). Note that the sending of a request may result in OperationInterruptedException before the request is even sent.

EventHandler TimedOut

Summary

This event is fired whenever Call() decides that a timeout has occurred while waiting for a reply from the service.
Remarks

See also OnTimedOut().

Constructor Detail

SimpleRpcClient

```
public SimpleRpcClient(IModel model, string exchange, string exchangeType, string
routingKey)
```

	Name	Type
Parameters	model	IModel
	exchange	string
	exchangeType	string
	routingKey	string

Summary

Construct an instance that will deliver to the named and typed exchange, with the given routing key.

SimpleRpcClient

```
public SimpleRpcClient(IModel model, PublicationAddress address)
```

	Name	Type
Parameters	model	IModel
	address	PublicationAddress

Summary

Construct an instance that will deliver to the given address.

SimpleRpcClient

```
public SimpleRpcClient(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Construct an instance with no configured Address. The Address property must be set before Call() or Cast() are called.

SimpleRpcClient

```
public SimpleRpcClient(IModel model, string queueName)
```

	Name	Type
Parameters	model	IModel
	queueName	string

Summary

Construct an instance that will deliver to the default exchange (""), with routing key equal to the passed in queueName, thereby delivering directly to a named queue on the AMQP server.

Method Detail

Call

```
public virtual byte[] Call(byte[] body)
```

Flags	public virtual
Return type	byte[]

	Name	Type
Parameters	body	byte[]

Summary

Sends a simple byte[] message, without any custom headers or properties.

Remarks

Delegates directly to Call(IBasicProperties, byte[]), and discards the properties of the received reply, returning only the body of the reply.

Calls OnTimedOut() and OnDisconnected() when a timeout or disconnection, respectively, is detected when waiting for our reply.

Returns null if the request timed out or if we were disconnected before a reply arrived.

The reply message, if any, is acknowledged to the AMQP server via Subscription.Ack().

Call

```
public virtual byte[] Call(IBasicProperties requestProperties, byte[] body, out
IBasicProperties replyProperties)
```

Flags	public virtual	
Return type	`byte[]`	
Parameters	**Name**	**Type**
	requestProperties	`IBasicProperties`
	body	`byte[]`
	replyProperties	`out IBasicProperties`

Summary

Sends a byte[] message and IBasicProperties header, returning both the body and headers of the received reply.

Remarks

Sets the "replyProperties" outbound parameter to the properties of the received reply, and returns the byte[] body of the reply.

Calls OnTimedOut() and OnDisconnected() when a timeout or disconnection, respectively, is detected when waiting for our reply.

Both sets "replyProperties" to null and returns null when either the request timed out or we were disconnected before a reply arrived.

The reply message, if any, is acknowledged to the AMQP server via Subscription.Ack().

Call

```
public virtual object[] Call(object[] args)
```

Flags	public virtual	
Return type	`object[]`	
Parameters	**Name**	**Type**
	args	`object[]`

Summary

Sends a "jms/stream-message"-encoded RPC request, and expects an RPC reply in the same format.

Remarks

The arguments passed in must be of types that are representable as JMS StreamMessage values, and so must the results returned from the service in its reply message.

Calls OnTimedOut() and OnDisconnected() when a timeout or disconnection, respectively, is detected when waiting for our reply.

Returns null if the request timed out or if we were disconnected before a reply arrived.

The reply message, if any, is acknowledged to the AMQP server via Subscription.Ack().

See

- RabbitMQ.Client.Content.IStreamMessageBuilder
- RabbitMQ.Client.Content.IStreamMessageReader

Call

```
public virtual BasicDeliverEventArgs Call(IBasicProperties requestProperties, byte[] body)
```

Flags	public virtual	
Return type	BasicDeliverEventArgs	
	Name	**Type**
Parameters	requestProperties	IBasicProperties
	body	byte[]

Summary

Sends a byte[]/IBasicProperties RPC request, returning full information about the delivered reply as a BasicDeliverEventArgs.

Remarks

This is the most general/lowest-level Call()-style method on SimpleRpcClient. It sets CorrelationId and ReplyTo on the request message's headers before transmitting the request to the service via the AMQP server. If the reply's CorrelationId does not match the request's CorrelationId, ProtocolViolationException will be thrown.

Calls OnTimedOut() and OnDisconnected() when a timeout or disconnection, respectively, is detected when waiting for our reply.

Returns null if the request timed out or if we were disconnected before a reply arrived.

The reply message, if any, is acknowledged to the AMQP server via Subscription.Ack().

See

- System.Net.ProtocolViolationException

Cast

```
public virtual void Cast(IBasicProperties requestProperties, byte[] body)
```

Flags	public virtual	
Return type	void	
	Name	**Type**
Parameters	requestProperties	IBasicProperties
	body	byte[]

Summary

Sends an asynchronous/one-way message to the service.

Close

```
public void Close()
```

Flags public
Return type `void`
Summary

Close the reply subscription associated with this instance, if any.
Remarks

Simply delegates to calling Subscription.Close(). Clears the Subscription property, so that subsequent Call()s, if any, will re-initialize it to a fresh Subscription instance.

OnDisconnected

```
public virtual void OnDisconnected()
```

Flags public virtual
Return type `void`
Summary

Signals that the Subscription we use for receiving our RPC replies was disconnected while we were waiting.
Remarks

Fires the Disconnected event.

OnTimedOut

```
public virtual void OnTimedOut()
```

Flags public virtual
Return type `void`
Summary

Signals that the configured timeout fired while waiting for an RPC reply.
Remarks

Fires the TimedOut event.
Index

Namespace RabbitMQ.Util

Summary

Internal. Utility classes.

Types

Type	Summary
BlockingCell	A thread-safe single-assignment reference cell.
DebugUtil	Miscellaneous debugging and development utilities.
Either	Models the disjoint union of two alternatives, a "left" alternative and a "right" alternative.
EitherAlternative	Used internally by class Either.
IntAllocator	(undocumented)
IntAllocator.IntervalList	(undocumented)
NetworkBinaryReader	Subclass of BinaryReader that reads integers etc in correct network order.
NetworkBinaryWriter	Subclass of BinaryWriter that writes integers etc in correct network order.
SharedQueue	A thread-safe shared queue implementation.
SharedQueue<T>	A thread-safe shared queue implementation.
SharedQueueEnumerator<T>	Implementation of the IEnumerator interface, for permitting SharedQueue to be used in foreach loops.
SynchronizedList<T>	(undocumented)
XmlUtil	Miscellaneous helpful XML utilities.

Index | Namespace RabbitMQ.Util

public class BlockingCell

Summary

A thread-safe single-assignment reference cell.

Remarks

A fresh BlockingCell holds no value (is empty). Any thread reading the Value property when the cell is empty will block until a value is made available by some other thread. The Value property can only be set once - on the first call, the BlockingCell is considered full, and made immutable. Further attempts to set Value result in a thrown InvalidOperationException.

Property Summary

Flags	Type	Name	Summary
public	`object`	`Value` (rw)	Retrieve the cell's value, blocking if none exists at present, or supply a value to an empty cell, thereby filling it.

Constructor Summary

Flags	Name	Summary
public	`BlockingCell()`	Construct an empty BlockingCell.

Method Summary

Flags	Name	Summary
public	`bool GetValue(int millisecondsTimeout, out object result)`	Retrieve the cell's value, waiting for the given timeout if no value is immediately available.
public static	`int validatedTimeout(int timeout)`	Return valid timeout value

Property Detail

public object Value (rw)

Summary

Retrieve the cell's value, blocking if none exists at present, or supply a value to an empty cell, thereby filling it.

Exception

Constructor Detail

BlockingCell

`public BlockingCell()`

Summary

Construct an empty BlockingCell.

Method Detail

GetValue

`public bool GetValue(int millisecondsTimeout, out object result)`

Flags	public
Return type	`bool`

	Name	Type
Parameters	millisecondsTimeout	`int`
	result	`out object`

Summary

Retrieve the cell's value, waiting for the given timeout if no value is immediately available.

Remarks

If a value is present in the cell at the time the call is made, the call will return immediately. Otherwise, the calling thread blocks until either a value appears, or millisecondsTimeout milliseconds have elapsed.

Returns true in the case that the value was available before the timeout, in which case the out parameter "result" is set to the value itself.

If no value was available before the timeout, returns false, and sets "result" to null.

A timeout of -1 (i.e. System.Threading.Timeout.Infinite) will be interpreted as a command to wait for an indefinitely long period of time for the cell's value to become available. See the MSDN documentation for System.Threading.Monitor.Wait(object,int).

validatedTimeout

```
public static int validatedTimeout(int timeout)
```

Flags	public static	
Return type	`int`	
Parameters	**Name**	**Type**
	timeout	`int`

Summary

Return valid timeout value

Remarks

If value of the parameter is less then zero, return 0 to mean infinity