RabbitMQ .NET client library API guide

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

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.ConnectionShutdownEventHandler	Delegate used to process connection shutdown notifications.
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 contro events.
RabbitMQ.Client.Events.ModelShutdownEventHandler	Delegate used to process model shutdown notifications.
RabbitMQ.Client.Exceptions.AlreadyClosedException	Thrown when the application tries to make use of a session or connection that has already been shut down.
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.SharedQueueEnumerator	Implementation of the IEnumerator interface, for permitting SharedQue to be used in foreach loops.
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)
```

Flags	public static	
Return type	AmqpTcpEndpoint[]	
	Name	**Type**
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

public struct AmqpTimestamp

- extends `ValueType`

Summary

Structure holding an AMQP timestamp, a posix 64-bit time_t.

Remarks

When converting between an AmqpTimestamp and a System.DateTime, be aware of the effect of your local timezone. In particular, different versions of the .NET framework assume different defaults.

We have chosen a signed 64-bit time_t here, since the AMQP specification through versions 0-9 is silent on whether timestamps are signed or unsigned.

Property Summary

Flags	Type	Name	Summary
public	`long`	`UnixTime` (r)	Retrieve the time_t from this structure.

Constructor Summary

Flags	Name	Summary
public	`AmqpTimestamp(long unixTime)`	Construct an AmqpTimestamp holding the given time_t value.

Method Summary

Flags	Name	Summary
public virtual	`string ToString()`	Provides a debugger-friendly display.

Property Detail

public long UnixTime (r)

Summary

Retrieve the time_t from this structure.

Constructor Detail

AmqpTimestamp

```
public AmqpTimestamp(long unixTime)
```

Parameters	Name	Type
	unixTime	`long`

Summary

Construct an AmqpTimestamp holding the given time_t value.

Remarks

Method Detail

ToString

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

Flags public virtual

Return type `string`

Summary

Provides a debugger-friendly display.

Index | Namespace RabbitMQ.Client

public class AmqpVersion

Summary

Represents a version of the AMQP specification.

Remarks

Vendor-specific variants of particular official specification versions exist: this class simply represents the AMQP specification version, and does not try to represent information about any custom variations involved.

AMQP version 0-8 peers sometimes advertise themselves as version 8-0: for this reason, this class's constructor special-cases 8-0, rewriting it at construction time to be 0-8 instead.

Property Summary

Flags	Type	Name	Summary
public	`int`	`Major` (r)	The AMQP specification major version number
public	`int`	`Minor` (r)	The AMQP specification minor version number

Constructor Summary

Flags	Name	Summary
public	`AmqpVersion(int major, int minor)`	Construct an AmqpVersion from major and minor version numbers.

Method Summary

Flags	Name	Summary
public virtual	`bool Equals(object other)`	Implement value-equality comparison.
public virtual	`int GetHashCode()`	Implement hashing as for value-equality.
public virtual	`string ToString()`	Format appropriately for display.

Property Detail

public int Major (r)

Summary

The AMQP specification major version number

public int Minor (r)

Summary

The AMQP specification minor version number

Constructor Detail

AmqpVersion

```
public AmqpVersion(int major, int minor)
```

	Name	Type
Parameters	major	`int`
	minor	`int`

Summary

Construct an AmqpVersion from major and minor version numbers.

Remarks

Converts major=8 and minor=0 into major=0 and minor=8. Please see the class comment.

Method Detail

Equals

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

Flags	public virtual
Return type	`bool`

Parameters	**Name**	**Type**
	other	`object`

Summary

Implement value-equality comparison.

GetHashCode

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

Flags	public virtual
Return type	`int`

Summary

Implement hashing as for value-equality.

ToString

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

Flags	public virtual
Return type	`string`

Summary

Format appropriately for display.

Remarks

The specification currently uses "MAJOR-MINOR" as a display format.

Index | Namespace RabbitMQ.Client

public interface AuthMechanism

Summary

A pluggable authentication mechanism.

Method Summary

Name	Summary
byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)	Handle one round of challenge-response

Method Detail

handleChallenge

```
byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)
```

Return type byte[]

	Name	Type
Parameters	challenge	byte[]
	factory	ConnectionFactory

Summary

Handle one round of challenge-response

Index | Namespace RabbitMQ.Client

public interface AuthMechanismFactory

Property Summary

Type	Name	Summary
`string`	Name (r)	The name of the authentication mechanism, as negotiated on the wire

Method Summary

Name	Summary
AuthMechanism GetInstance()	Return a new authentication mechanism implementation

Property Detail

string Name (r)

Summary

The name of the authentication mechanism, as negotiated on the wire

Method Detail

GetInstance

```
AuthMechanism GetInstance()
```

Return type AuthMechanism

Summary

Return a new authentication mechanism implementation

Index | Namespace RabbitMQ.Client

public class BasicGetResult

Summary

Represents Basic.GetOk responses from the server.

Remarks

Basic.Get either returns an instance of this class, or null if a Basic.GetEmpty was received.

Property Summary

Flags	Type	Name	Summary
public	IBasicProperties	BasicProperties (r)	Retrieves the Basic-class content header properties for this message.
public	byte[]	Body (r)	Retrieves the body of this message.
public	ulong	DeliveryTag (r)	Retrieve the delivery tag for this message. See also IModel.BasicAck.
public	string	Exchange (r)	Retrieve the exchange this message was published to.
public	uint	MessageCount (r)	Retrieve the number of messages pending on the queue, excluding the message being delivered.
public	bool	Redelivered (r)	Retrieve the redelivered flag for this message.
public	string	RoutingKey (r)	Retrieve the routing key with which this message was published.

Constructor Summary

Flags	Name	Summary
public	BasicGetResult(ulong deliveryTag, bool redelivered, string exchange, string routingKey, uint messageCount, IBasicProperties basicProperties, byte[] body)	Sets the new instance's properties from the arguments passed in.

Property Detail

public IBasicProperties BasicProperties (r)

Summary

Retrieves the Basic-class content header properties for this message.

public byte[] Body (r)

Summary

Retrieves the body of this message.

public ulong DeliveryTag (r)

Summary

Retrieve the delivery tag for this message. See also IModel.BasicAck.

public string Exchange (r)

Summary

Retrieve the exchange this message was published to.

public uint MessageCount (r)

Summary

Retrieve the number of messages pending on the queue, excluding the message being delivered.

Remarks

Note that this figure is indicative, not reliable, and can change arbitrarily as messages are added to the queue and removed by other clients.

public bool Redelivered (r)

Summary

Retrieve the redelivered flag for this message.

public string RoutingKey (r)

Summary

Retrieve the routing key with which this message was published.

Constructor Detail

BasicGetResult

```
public BasicGetResult(ulong deliveryTag, bool redelivered, string exchange, string
routingKey, uint messageCount, IBasicProperties basicProperties, byte[] body)
```

Parameters

Name	Type
deliveryTag	ulong
redelivered	bool
exchange	string
routingKey	string
messageCount	uint
basicProperties	IBasicProperties
body	byte[]

Summary

Sets the new instance's properties from the arguments passed in.

Index | Namespace RabbitMQ.Client

public class BinaryTableValue

Summary

Wrapper for a byte[]. May appear as values read from and written to AMQP field tables.

Remarks

The sole reason for the existence of this class is to permit encoding of byte[] as 'x' in AMQP field tables, an extension to the specification that is part of the tentative JMS mapping implemented by QPid.

Instances of this object may be found as values held in IDictionary instances returned from RabbitMQ.Client.Impl.WireFormatting.ReadTable, e.g. as part of IBasicProperties.Headers tables. Likewise, instances may be set as values in an IDictionary table to be encoded by RabbitMQ.Client.Impl.WireFormatting.WriteTable.

When an instance of this class is encoded/decoded, the type tag 'x' is used in the on-the-wire representation. The AMQP standard type tag 'S' is decoded to a raw byte[], and a raw byte[] is encoded as 'S'. Instances of System.String are converted to a UTF-8 binary representation, and then encoded using tag 'S'. In order to force the use of tag 'x', instances of this class must be used.

Property Summary

Flags	Type	Name	Summary
public	`byte[]`	Bytes (rw)	The wrapped byte array, as decoded or as to be encoded.

Constructor Summary

Flags	Name	Summary
public	BinaryTableValue(byte[] bytes)	Constructs an instance with the passed-in value for its Bytes property.
public	BinaryTableValue()	Constructs an instance with null for its Bytes property.

Property Detail

public byte[] Bytes (rw)

Summary

The wrapped byte array, as decoded or as to be encoded.

Constructor Detail

BinaryTableValue

```
public BinaryTableValue(byte[] bytes)
```

Parameters	Name	Type
	bytes	`byte[]`

Summary

Constructs an instance with the passed-in value for its Bytes property.

BinaryTableValue

```
public BinaryTableValue()
```

Summary

Constructs an instance with null for its Bytes property.
Index | Namespace RabbitMQ.Client

public class ConnectionFactory

Nested types: ObtainSocket

Summary

Main entry point to the RabbitMQ .NET AMQP client API. Constructs IConnection instances.

Remarks

A simple example of connecting to a broker:

```
ConnectionFactory factory = new ConnectionFactory();
//
// The next six lines are optional:
factory.UserName = ConnectionFactory.DefaultUser;
factory.Password = ConnectionFactory.DefaultPass;
factory.VirtualHost = ConnectionFactory.DefaultVHost;
factory.Protocol = Protocols.FromEnvironment();
factory.HostName = hostName;
factory.Port     = AmqpTcpEndpoint.UseDefaultPort;
//
IConnection conn = factory.CreateConnection();
//
IModel ch = conn.CreateModel();
//
// ... use ch's IModel methods ...
//
ch.Close(Constants.ReplySuccess, "Closing the channel");
conn.Close(Constants.ReplySuccess, "Closing the connection");
```

The same example, written more compactly with AMQP URIs:

```
ConnectionFactory factory = new ConnectionFactory();
factory.Uri = "amqp://localhost";
IConnection conn = factory.CreateConnection();
...
```

Please see also the API overview and tutorial in the User Guide.

Note that the Uri property takes a string representation of an AMQP URI. Omitted URI parts will take default values. The host part of the URI cannot be omitted and URIs of the form "amqp://foo/" (note the trailling slash) also represent the default virtual host. The latter issue means that virtual hosts with an empty name are not addressable.

Field Summary

Flags	Type	Name	Summary
public	AuthMechanismFactory[]	AuthMechanisms	SASL auth mechanisms to use.
public	IDictionary	ClientProperties	Dictionary of client properties to be sent to the server
public static	AuthMechanismFactory[]	DefaultAuthMechanisms	Default SASL auth mechanisms to use.
public const	ushort	DefaultChannelMax	Default value for the desired maximum channel number, with zero meaning unlimited (value: 0)
public const	int	DefaultConnectionTimeout	Default value for connection attempt timeout, in milliseconds
public const	uint	DefaultFrameMax	Default value for the desired maximum frame size, with zero meaning unlimited (value: 0)
	ushort	DefaultHeartbeat	

public const			Default value for desired heartbeat interval, in seconds, with zero meaning none (value: 0)
public const	string	DefaultPass	Default password (value: "guest")
public const	string	DefaultUser	Default user name (value: "guest")
public const	string	DefaultVHost	Default virtual host (value: "/")
public	string	HostName	The host to connect to
public	string	Password	Password to use when authenticating to the server
public	int	Port	The port to connect on. AmqpTcpEndpoint.UseDefaultPort indicates the default for the protocol should be used.
public	IProtocol	Protocol	The AMQP protocol to be used
public	ushort	RequestedChannelMax	Maximum channel number to ask for
public	int	RequestedConnectionTimeout	Timeout setting for connection attempts (in milliseconds)
public	uint	RequestedFrameMax	Frame-max parameter to ask for (in bytes)
public	ushort	RequestedHeartbeat	Heartbeat setting to request (in seconds)
public	ConnectionFactory.ObtainSocket	SocketFactory	Set custom socket options by providing a SocketFactory
public	SslOption	Ssl	Ssl options setting
public	string	UserName	Username to use when authenticating to the server
public	string	VirtualHost	Virtual host to access during this connection

Property Summary

Flags	Type	Name	Summary
public	AmqpTcpEndpoint	Endpoint (rw)	The AMQP connection target
public	Uri	uri (w)	Set connection parameters using the amqp or amqps scheme
public	string	Uri (w)	Set connection parameters using the amqp or amqps scheme

Constructor Summary

Flags	Name	Summary
public	ConnectionFactory()	Construct a fresh instance, with all fields set to their respective defaults.

Method Summary

Flags	Name	Summary
public	AuthMechanismFactory AuthMechanismFactory(string[] mechs)	Given a list of mechanism names supported by the server, select a preferred mechanism, or null if we have none in common.
public virtual	IConnection CreateConnection()	Create a connection to the specified endpoint No broker-originated redirects are permitted.
public virtual	IConnection CreateConnection(int maxRedirects)	Create a connection to the first available endpoint in the list provided. Up to a maximum of maxRedirects broker-originated redirects are permitted for each endpoint tried.

public static	TcpClient DefaultSocketFactory(AddressFamily addressFamily) (undocumented)

Field Detail

public AuthMechanismFactory[] AuthMechanisms

Summary

SASL auth mechanisms to use.

public IDictionary ClientProperties

Summary

Dictionary of client properties to be sent to the server

public static AuthMechanismFactory[] DefaultAuthMechanisms

Summary

Default SASL auth mechanisms to use.

public const ushort DefaultChannelMax

Summary

Default value for the desired maximum channel number, with zero meaning unlimited (value: 0)

public const int DefaultConnectionTimeout

Summary

Default value for connection attempt timeout, in milliseconds

public const uint DefaultFrameMax

Summary

Default value for the desired maximum frame size, with zero meaning unlimited (value: 0)

public const ushort DefaultHeartbeat

Summary

Default value for desired heartbeat interval, in seconds, with zero meaning none (value: 0)

public const string DefaultPass

Summary

Default password (value: "guest")

public const string DefaultUser

Summary

Default user name (value: "guest")

public const string DefaultVHost

Summary

Default virtual host (value: "/")

public string HostName

Summary

The host to connect to

public string Password

Summary

Password to use when authenticating to the server

public int Port

Summary

The port to connect on. AmqpTcpEndpoint.UseDefaultPort indicates the default for the protocol should be used.

public IProtocol Protocol

Summary

The AMQP protocol to be used

public ushort RequestedChannelMax

Summary

Maximum channel number to ask for

public int RequestedConnectionTimeout

Summary

Timeout setting for connection attempts (in milliseconds)

public uint RequestedFrameMax

Summary

Frame-max parameter to ask for (in bytes)

public ushort RequestedHeartbeat

Summary

Heartbeat setting to request (in seconds)

public ConnectionFactory.ObtainSocket SocketFactory

Summary

Set custom socket options by providing a SocketFactory

public SslOption Ssl

Summary

Ssl options setting

public string UserName

Summary

Username to use when authenticating to the server

public string VirtualHost

Summary

Virtual host to access during this connection

Property Detail

public AmqpTcpEndpoint Endpoint (rw)

Summary

The AMQP connection target

public Uri uri (w)

Summary

Set connection parameters using the amqp or amqps scheme

public string Uri (w)

Summary

Set connection parameters using the amqp or amqps scheme

Constructor Detail

ConnectionFactory

```
public ConnectionFactory()
```

Summary

Construct a fresh instance, with all fields set to their respective defaults.

Method Detail

AuthMechanismFactory

```
public AuthMechanismFactory AuthMechanismFactory(string[] mechs)
```

Flags public
Return type AuthMechanismFactory

Parameters	Name	Type
	mechs	string[]

Summary

Given a list of mechanism names supported by the server, select a preferred mechanism, or null if we have none in common.

CreateConnection

```
public virtual IConnection CreateConnection()
```

Flags	public virtual
Return type	IConnection

Summary

Create a connection to the specified endpoint No broker-originated redirects are permitted.

CreateConnection

```
public virtual IConnection CreateConnection(int maxRedirects)
```

Flags	public virtual
Return type	IConnection

Parameters	Name	Type
	maxRedirects	int

Summary

Create a connection to the first available endpoint in the list provided. Up to a maximum of maxRedirects broker-originated redirects are permitted for each endpoint tried.

DefaultSocketFactory

```
public static TcpClient DefaultSocketFactory(AddressFamily addressFamily)
```

Flags	public static
Return type	TcpClient

Parameters	Name	Type
	addressFamily	AddressFamily

Index | Namespace RabbitMQ.Client

public class DefaultBasicConsumer

- implements IBasicConsumer

Summary

Useful default/base implementation of IBasicConsumer. Subclass and override HandleBasicDeliver in application code.

Remarks

Note that the "Handle*" methods run in the connection's thread! Consider using QueueingBasicConsumer, which uses a SharedQueue instance to safely pass received messages across to user threads, or RabbitMQ.Client.MessagePatterns.Subscription, which manages resource declaration and binding in addition to providing a thread-safe interface.

Property Summary

Flags	Type	Name	Summary
public	string	ConsumerTag (rw)	Retrieve the consumer tag this consumer is registered as; to be used when discussing this consumer with the server, for instance with IModel.BasicCancel().
public	bool	IsRunning (r)	Returns true while the consumer is registered and expecting deliveries from the broker.
public virtual final	IModel	Model (rw)	Retrieve the IModel instance this consumer is registered with.
public	ShutdownEventArgs	ShutdownReason (r)	If our IModel shuts down, this property will contain a description of the reason for the shutdown. Otherwise it will contain null. See ShutdownEventArgs.

Constructor Summary

Flags	Name	Summary
public	DefaultBasicConsumer(IModel model)	Constructor which sets the Model property to the given value.
public	DefaultBasicConsumer()	Default constructor.

Method Summary

Flags	Name	Summary
public virtual	void HandleBasicCancel(string consumerTag)	Default implementation - calls OnCancel().
public virtual	void HandleBasicCancelOk(string consumerTag)	Default implementation - calls OnCancel().
public virtual	void HandleBasicConsumeOk(string consumerTag)	Default implementation - sets the ConsumerTag property and sets IsRunning to true.
public virtual	void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool redelivered, string exchange, string routingKey, IBasicProperties properties, byte[] body)	Default implementation - override in subclasses.
public virtual	void HandleModelShutdown(IModel model, ShutdownEventArgs reason)	Default implementation - sets ShutdownReason and calls OnCancel().
public virtual	void OnCancel()	Default implementation - overridable in subclasses.

Property Detail

public string ConsumerTag (rw)

Summary

Retrieve the consumer tag this consumer is registered as; to be used when discussing this consumer with the server, for instance with IModel.BasicCancel().

public bool IsRunning (r)

Summary

Returns true while the consumer is registered and expecting deliveries from the broker.

public virtual final IModel Model (rw)

Summary

Retrieve the IModel instance this consumer is registered with.

public ShutdownEventArgs ShutdownReason (r)

Summary

If our IModel shuts down, this property will contain a description of the reason for the shutdown. Otherwise it will contain null. See ShutdownEventArgs.

Constructor Detail

DefaultBasicConsumer

```
public DefaultBasicConsumer(IModel model)
```

Parameters	Name	Type
	model	IModel

Summary

Constructor which sets the Model property to the given value.

DefaultBasicConsumer

```
public DefaultBasicConsumer()
```

Summary

Default constructor.

Method Detail

HandleBasicCancel

```
public virtual void HandleBasicCancel(string consumerTag)
```

Flags	public virtual	
Return type	void	
Parameters	**Name**	**Type**

consumerTag `string`

Summary

Default implementation - calls OnCancel().

HandleBasicCancelOk

```
public virtual void HandleBasicCancelOk(string consumerTag)
```

Flags	public virtual
Return type	`void`

Parameters

Name	Type
consumerTag	`string`

Summary

Default implementation - calls OnCancel().

HandleBasicConsumeOk

```
public virtual void HandleBasicConsumeOk(string consumerTag)
```

Flags	public virtual
Return type	`void`

Parameters

Name	Type
consumerTag	`string`

Summary

Default implementation - sets the ConsumerTag property and sets IsRunning to true.

HandleBasicDeliver

```
public virtual void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool
redelivered, string exchange, string routingKey, IBasicProperties properties, byte[]
body)
```

Flags	public virtual
Return type	`void`

Parameters

Name	Type
consumerTag	`string`
deliveryTag	`ulong`
redelivered	`bool`
exchange	`string`
routingKey	`string`
properties	`IBasicProperties`
body	`byte[]`

Summary

Default implementation - override in subclasses.

Remarks

Does nothing with the passed in information. Note that in particular, some delivered messages may require acknowledgement via IModel.BasicAck; the implementation of this method in this class does NOT acknowledge such messages.

HandleModelShutdown

```
public virtual void HandleModelShutdown(IModel model, ShutdownEventArgs reason)
```

Flags	public virtual	
Return type	void	
	Name	**Type**
Parameters	model	IModel
	reason	ShutdownEventArgs

Summary

Default implementation - sets ShutdownReason and calls OnCancel().

OnCancel

```
public virtual void OnCancel()
```

Flags	public virtual
Return type	void

Summary

Default implementation - overridable in subclasses.

Remarks

This default implementation simply sets the IsRunning property to false, and takes no further action.

Index | Namespace RabbitMQ.Client

public class ExchangeType

Summary

Convenience class providing compile-time names for standard exchange types.

Remarks

Use the static members of this class as values for the "exchangeType" arguments for IModel methods such as ExchangeDeclare. The broker may be extended with additional exchange types that do not appear in this class.

Field Summary

Flags	Type	Name	Summary
public const	`string`	`Direct`	Exchange type used for AMQP direct exchanges.
public const	`string`	`Fanout`	Exchange type used for AMQP fanout exchanges.
public const	`string`	`Headers`	Exchange type used for AMQP headers exchanges.
public const	`string`	`Topic`	Exchange type used for AMQP topic exchanges.

Method Summary

Flags	Name	Summary
public static	`ICollection All()`	Retrieve a collection containing all standard exchange types.

Field Detail

public const string Direct

Summary

Exchange type used for AMQP direct exchanges.

public const string Fanout

Summary

Exchange type used for AMQP fanout exchanges.

public const string Headers

Summary

Exchange type used for AMQP headers exchanges.

public const string Topic

Summary

Exchange type used for AMQP topic exchanges.

Method Detail

All

```
public static ICollection All()
```

Flags public static

Return type `ICollection`

Summary

Retrieve a collection containing all standard exchange types.
Index | Namespace RabbitMQ.Client

public class ExternalMechanism

- implements AuthMechanism

Constructor Summary

Flags	Name	Summary
public	ExternalMechanism()	(undocumented)

Method Summary

Flags	Name	Summary
public virtual final	byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)	(undocumented)

Constructor Detail

ExternalMechanism

```
public ExternalMechanism()
```

Method Detail

handleChallenge

```
public virtual final byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)
```

Flags	public virtual final	
Return type	byte[]	
	Name	**Type**
Parameters	challenge	byte[]
	factory	ConnectionFactory

Index | Namespace RabbitMQ.Client

public class ExternalMechanismFactory

- implements AuthMechanismFactory

Property Summary

Flags	Type	Name	Summary
public virtual final	string	Name (r)	(undocumented)

Constructor Summary

Flags	Name	Summary
public	ExternalMechanismFactory()	(undocumented)

Method Summary

Flags	Name	Summary
public virtual final	AuthMechanism GetInstance()	(undocumented)

Property Detail

public virtual final string Name (r)

Constructor Detail

ExternalMechanismFactory

```
public ExternalMechanismFactory()
```

Method Detail

GetInstance

```
public virtual final AuthMechanism GetInstance()
```

Flags	public virtual final
Return type	AuthMechanism

Index | Namespace RabbitMQ.Client

public interface IBasicConsumer

Summary

Consumer interface for Basic content-class. Used to receive messages from a queue by subscription.
Remarks

See IModel.BasicConsume, IModel.BasicCancel.

Note that the "Handle*" methods run in the connection's thread! Consider using QueueingBasicConsumer, which uses a SharedQueue instance to safely pass received messages across to user threads.

Property Summary

Type	Name	Summary
IModel	Model (r)	Retrieve the IModel this consumer is associated with, for use in acknowledging received messages, for instance.

Method Summary

Name	Summary
void HandleBasicCancel(string consumerTag)	Called when the consumer is cancelled for reasons other than by a basicCancel: e.g. the queue has been deleted (either by this channel or by any other channel). See handleCancelOk for notification of consumer cancellation due to basicCancel.
void HandleBasicCancelOk(string consumerTag)	Called upon successful deregistration of the consumer from the broker.
void HandleBasicConsumeOk(string consumerTag)	Called upon successful registration of the consumer with the broker.
void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool redelivered, string exchange, string routingKey, IBasicProperties properties, byte[] body)	Called each time a message arrives for this consumer.
void HandleModelShutdown(IModel model, ShutdownEventArgs reason)	Called when the model shuts down.

Property Detail

IModel Model (r)

Summary

Retrieve the IModel this consumer is associated with, for use in acknowledging received messages, for instance.

Method Detail

HandleBasicCancel

```
void HandleBasicCancel(string consumerTag)
```

Return type void

Parameters

Name	Type
consumerTag	string

Summary

Called when the consumer is cancelled for reasons other than by a basicCancel: e.g. the queue has been deleted (either by this channel or by any other channel). See handleCancelOk for notification of consumer cancellation due to basicCancel.

HandleBasicCancelOk

```
void HandleBasicCancelOk(string consumerTag)
```

Return type `void`

Parameters

Name	Type
consumerTag	`string`

Summary

Called upon successful deregistration of the consumer from the broker.

HandleBasicConsumeOk

```
void HandleBasicConsumeOk(string consumerTag)
```

Return type `void`

Parameters

Name	Type
consumerTag	`string`

Summary

Called upon successful registration of the consumer with the broker.

HandleBasicDeliver

```
void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool redelivered, string
exchange, string routingKey, IBasicProperties properties, byte[] body)
```

Return type `void`

Parameters

Name	Type
consumerTag	`string`
deliveryTag	`ulong`
redelivered	`bool`
exchange	`string`
routingKey	`string`
properties	`IBasicProperties`
body	`byte[]`

Summary

Called each time a message arrives for this consumer.

Remarks

Be aware that acknowledgement may be required. See IModel.BasicAck.

HandleModelShutdown

```
void HandleModelShutdown(IModel model, ShutdownEventArgs reason)
```

Return type `void`

	Name	Type
Parameters	model	IModel
	reason	ShutdownEventArgs

Summary

Called when the model shuts down.
Index | Namespace RabbitMQ.Client

public interface IBasicProperties

- implements ICloneable
- implements IContentHeader

Summary

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.

Remarks

The specification code generator provides protocol-version-specific implementations of this interface. To obtain an implementation of this interface in a protocol-version-neutral way, use IModel.CreateBasicProperties().

Each property is readable, writable and clearable: a cleared property will not be transmitted over the wire. Properties on a fresh instance are clear by default.

Property Summary

Type	Name	Summary
string	AppId (rw)	creating application id
string	ClusterId (rw)	intra-cluster routing identifier (cluster id is deprecated in AMQP 0-9-1)
string	ContentEncoding (rw)	MIME content encoding
string	ContentType (rw)	MIME content type
string	CorrelationId (rw)	application correlation identifier
byte	DeliveryMode (rw)	non-persistent (1) or persistent (2)
string	Expiration (rw)	message expiration specification
IDictionary	Headers (rw)	message header field table
string	MessageId (rw)	application message identifier
byte	Priority (rw)	message priority, 0 to 9
string	ReplyTo (rw)	destination to reply to
PublicationAddress	ReplyToAddress (rw)	Convenience property; parses ReplyTo property using PublicationAddress.Parse, and serializes it using PublicationAddress.ToString. Returns null if ReplyTo property cannot be parsed by PublicationAddress.Parse.
AmqpTimestamp	Timestamp (rw)	message timestamp
string	Type (rw)	message type name
string	UserId (rw)	creating user id

Method Summary

Name	Summary
void ClearAppId()	Clear the AppId property.
void ClearClusterId()	Clear the ClusterId property. (cluster id is deprecated in AMQP 0-9-1)
void ClearContentEncoding()	Clear the ContentEncoding property.
void ClearContentType()	Clear the ContentType property.
void ClearCorrelationId()	Clear the CorrelationId property.
void ClearDeliveryMode()	Clear the DeliveryMode property.
void ClearExpiration()	Clear the Expiration property.

void ClearHeaders()	Clear the Headers property.
void ClearMessageId()	Clear the MessageId property.
void ClearPriority()	Clear the Priority property.
void ClearReplyTo()	Clear the ReplyTo property.
void ClearTimestamp()	Clear the Timestamp property.
void ClearType()	Clear the Type property.
void ClearUserId()	Clear the UserId property.
bool IsAppIdPresent()	Returns true iff the AppId property is present.
bool IsClusterIdPresent()	Returns true iff the ClusterId property is present. (cluster id is deprecated in AMQP 0-9-1)
bool IsContentEncodingPresent()	Returns true iff the ContentEncoding property is present.
bool IsContentTypePresent()	Returns true iff the ContentType property is present.
bool IsCorrelationIdPresent()	Returns true iff the CorrelationId property is present.
bool IsDeliveryModePresent()	Returns true iff the DeliveryMode property is present.
bool IsExpirationPresent()	Returns true iff the Expiration property is present.
bool IsHeadersPresent()	Returns true iff the Headers property is present.
bool IsMessageIdPresent()	Returns true iff the MessageId property is present.
bool IsPriorityPresent()	Returns true iff the Priority property is present.
bool IsReplyToPresent()	Returns true iff the ReplyTo property is present.
bool IsTimestampPresent()	Returns true iff the Timestamp property is present.
bool IsTypePresent()	Returns true iff the Type property is present.
bool IsUserIdPresent()	Returns true iff the UserId property is present.
void SetPersistent(bool persistent)	Sets DeliveryMode to either persistent (2) or non-persistent (1).

Property Detail

string AppId (rw)

Summary

creating application id

string ClusterId (rw)

Summary

intra-cluster routing identifier (cluster id is deprecated in AMQP 0-9-1)

string ContentEncoding (rw)

Summary

MIME content encoding

string ContentType (rw)

Summary

MIME content type

string CorrelationId (rw)

Summary

application correlation identifier

byte DeliveryMode (rw)

Summary

non-persistent (1) or persistent (2)

string Expiration (rw)

Summary

message expiration specification

IDictionary Headers (rw)

Summary

message header field table

string MessageId (rw)

Summary

application message identifier

byte Priority (rw)

Summary

message priority, 0 to 9

string ReplyTo (rw)

Summary

destination to reply to

PublicationAddress ReplyToAddress (rw)

Summary

Convenience property; parses ReplyTo property using PublicationAddress.Parse, and serializes it using PublicationAddress.ToString. Returns null if ReplyTo property cannot be parsed by PublicationAddress.Parse.

AmqpTimestamp Timestamp (rw)

Summary

message timestamp

string Type (rw)

Summary

message type name

string UserId (rw)

Summary

creating user id

Method Detail

ClearAppId

```
void ClearAppId()
```

Return type void
Summary

Clear the AppId property.

ClearClusterId

```
void ClearClusterId()
```

Return type void
Summary

Clear the ClusterId property. (cluster id is deprecated in AMQP 0-9-1)

ClearContentEncoding

```
void ClearContentEncoding()
```

Return type void
Summary

Clear the ContentEncoding property.

ClearContentType

```
void ClearContentType()
```

Return type void
Summary

Clear the ContentType property.

ClearCorrelationId

```
void ClearCorrelationId()
```

Return type void
Summary

Clear the CorrelationId property.

ClearDeliveryMode

```
void ClearDeliveryMode()
```

Return type void

Summary

Clear the DeliveryMode property.

ClearExpiration

```
void ClearExpiration()
```

Return type void
Summary

Clear the Expiration property.

ClearHeaders

```
void ClearHeaders()
```

Return type void
Summary

Clear the Headers property.

ClearMessageId

```
void ClearMessageId()
```

Return type void
Summary

Clear the MessageId property.

ClearPriority

```
void ClearPriority()
```

Return type void
Summary

Clear the Priority property.

ClearReplyTo

```
void ClearReplyTo()
```

Return type void
Summary

Clear the ReplyTo property.

ClearTimestamp

```
void ClearTimestamp()
```

Return type void
Summary

Clear the Timestamp property.

ClearType

```
void ClearType()
```

Return type `void`
Summary

Clear the Type property.

ClearUserId

```
void ClearUserId()
```

Return type `void`
Summary

Clear the UserId property.

IsAppIdPresent

```
bool IsAppIdPresent()
```

Return type `bool`
Summary

Returns true iff the AppId property is present.

IsClusterIdPresent

```
bool IsClusterIdPresent()
```

Return type `bool`
Summary

Returns true iff the ClusterId property is present. (cluster id is deprecated in AMQP 0-9-1)

IsContentEncodingPresent

```
bool IsContentEncodingPresent()
```

Return type `bool`
Summary

Returns true iff the ContentEncoding property is present.

IsContentTypePresent

```
bool IsContentTypePresent()
```

Return type `bool`
Summary

Returns true iff the ContentType property is present.

IsCorrelationIdPresent

```
bool IsCorrelationIdPresent()
```

Return type `bool`

Summary

Returns true iff the CorrelationId property is present.

IsDeliveryModePresent

```
bool IsDeliveryModePresent()
```

Return type bool
Summary

Returns true iff the DeliveryMode property is present.

IsExpirationPresent

```
bool IsExpirationPresent()
```

Return type bool
Summary

Returns true iff the Expiration property is present.

IsHeadersPresent

```
bool IsHeadersPresent()
```

Return type bool
Summary

Returns true iff the Headers property is present.

IsMessageIdPresent

```
bool IsMessageIdPresent()
```

Return type bool
Summary

Returns true iff the MessageId property is present.

IsPriorityPresent

```
bool IsPriorityPresent()
```

Return type bool
Summary

Returns true iff the Priority property is present.

IsReplyToPresent

```
bool IsReplyToPresent()
```

Return type bool
Summary

Returns true iff the ReplyTo property is present.

IsTimestampPresent

```
bool IsTimestampPresent()
```

Return type bool

Summary

Returns true iff the Timestamp property is present.

IsTypePresent

```
bool IsTypePresent()
```

Return type bool

Summary

Returns true iff the Type property is present.

IsUserIdPresent

```
bool IsUserIdPresent()
```

Return type bool

Summary

Returns true iff the UserId property is present.

SetPersistent

```
void SetPersistent(bool persistent)
```

Return type void

Parameters

Name	Type
persistent	bool

Summary

Sets DeliveryMode to either persistent (2) or non-persistent (1).

Remarks

The numbers 1 and 2 for delivery mode are "magic" in that they appear in the AMQP 0-8 and 0-9 specifications as part of the definition of the DeliveryMode Basic-class property, without being defined as named constants.

Calling this method causes DeliveryMode to take on a value. In order to reset DeliveryMode to the default empty condition, call ClearDeliveryMode.

Index | Namespace RabbitMQ.Client

public interface IConnection

- implements `IDisposable`

Summary

Main interface to an AMQP connection.

Remarks

Instances of IConnection are used to create fresh sessions/channels. The ConnectionFactory class is used to construct IConnection instances. Please see the documentation for ConnectionFactory for an example of usage. Alternatively, an API tutorial can be found in the User Guide.

Extends the IDisposable interface, so that the "using" statement can be used to scope the lifetime of a channel when appropriate.

Property Summary

Type	Name	Summary
`bool`	`AutoClose` (rw)	If true, will close the whole connection as soon as there are no channels open on it; if false, manual connection closure will be required.
`ushort`	`ChannelMax` (r)	The maximum channel number this connection supports (0 if unlimited). Usable channel numbers range from 1 to this number, inclusive.
`IDictionary`	`ClientProperties` (r)	A copy of the client properties that has been sent to the server.
`ShutdownEventArgs`	`CloseReason` (r)	Returns null if the connection is still in a state where it can be used, or the cause of its closure otherwise.
`AmqpTcpEndpoint`	`Endpoint` (r)	Retrieve the endpoint this connection is connected to.
`uint`	`FrameMax` (r)	The maximum frame size this connection supports (0 if unlimited).
`ushort`	`Heartbeat` (r)	The current heartbeat setting for this connection (0 for disabled), in seconds.
`bool`	`IsOpen` (r)	Returns true if the connection is still in a state where it can be used. Identical to checking if CloseReason == null.
`AmqpTcpEndpoint[]`	`KnownHosts` (r)	Returns the known hosts that came back from the broker in the connection.open-ok method at connection startup time. Null until the connection is completely open and ready for use.
`IProtocol`	`Protocol` (r)	The IProtocol this connection is using to communicate with its peer.
`IDictionary`	`ServerProperties` (r)	A dictionary of the server properties sent by the server while establishing the connection. This typically includes the product name and version of the server.
`IList`	`ShutdownReport` (r)	Returns the list of ShutdownReportEntry objects that contain information about any errors reported while closing the connection in the order they appeared

Event Summary

Type	Name	Summary
`CallbackExceptionEventHandler`	`CallbackException`	Signalled when an exception occurs in a callback invoked by the connection.
`ConnectionShutdownEventHandler`	`ConnectionShutdown`	Raised when the connection is destroyed.

Method Summary

Name	Summary
void Abort(ushort reasonCode, string reasonText)	Abort this connection and all its channels.
void Abort(int timeout)	Abort this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.
void Abort()	Abort this connection and all its channels.
void Abort(ushort reasonCode, string reasonText, int timeout)	Abort this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.
void Close(ushort reasonCode, string reasonText)	Close this connection and all its channels.
void Close()	Close this connection and all its channels.
void Close(int timeout)	Close this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.
void Close(ushort reasonCode, string reasonText, int timeout)	Close this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.
IModel CreateModel()	Create and return a fresh channel, session, and model.

Property Detail

bool AutoClose (rw)

Summary

If true, will close the whole connection as soon as there are no channels open on it; if false, manual connection closure will be required.

Remarks

Don't set AutoClose to true before opening the first channel, because the connection will be immediately closed if you do!

ushort ChannelMax (r)

Summary

The maximum channel number this connection supports (0 if unlimited). Usable channel numbers range from 1 to this number, inclusive.

IDictionary ClientProperties (r)

Summary

A copy of the client properties that has been sent to the server.

ShutdownEventArgs CloseReason (r)

Summary

Returns null if the connection is still in a state where it can be used, or the cause of its closure otherwise.

Remarks

Applications should use the ConnectionShutdown event to avoid race conditions. The scenario to avoid is checking CloseReason, seeing it is null (meaning the IConnection was available for use at the time of the check), and interpreting this mistakenly as a guarantee that the IConnection will remain usable for a time. Instead, the operation of interest should simply be attempted: if the IConnection is not in a usable state, an exception will be thrown (most likely OperationInterruptedException, but may vary depending on the particular operation being attempted).

AmqpTcpEndpoint Endpoint (r)

Summary

Retrieve the endpoint this connection is connected to.

uint FrameMax (r)

Summary

The maximum frame size this connection supports (0 if unlimited).

ushort Heartbeat (r)

Summary

The current heartbeat setting for this connection (0 for disabled), in seconds.

bool IsOpen (r)

Summary

Returns true if the connection is still in a state where it can be used. Identical to checking if CloseReason == null.

AmqpTcpEndpoint[] KnownHosts (r)

Summary

Returns the known hosts that came back from the broker in the connection.open-ok method at connection startup time. Null until the connection is completely open and ready for use.

IProtocol Protocol (r)

Summary

The IProtocol this connection is using to communicate with its peer.

IDictionary ServerProperties (r)

Summary

A dictionary of the server properties sent by the server while establishing the connection. This typically includes the product name and version of the server.

IList ShutdownReport (r)

Summary

Returns the list of ShutdownReportEntry objects that contain information about any errors reported while closing the connection in the order they appeared

Event Detail

CallbackExceptionEventHandler CallbackException

Summary

Signalled when an exception occurs in a callback invoked by the connection.

Remarks

This event is signalled when a ConnectionShutdown handler throws an exception. If, in future, more events appear on IConnection, then this event will be signalled whenever one of those event handlers throws an exception, as well.

ConnectionShutdownEventHandler ConnectionShutdown

Summary

Raised when the connection is destroyed.

Remarks

If the connection is already destroyed at the time an event handler is added to this event, the event handler will be fired immediately.

Method Detail

Abort

```
void Abort(ushort reasonCode, string reasonText)
```

Return type `void`

	Name	Type
Parameters	reasonCode	`ushort`
	reasonText	`string`

Summary

Abort this connection and all its channels.

Remarks

The method behaves in the same way as Abort(), with the only difference that the connection is closed with the given connection close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the connection

Abort

```
void Abort(int timeout)
```

Return type `void`

	Name	Type
Parameters	timeout	`int`

Summary

Abort this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.

Remarks

This method, behaves in a similar way as method Abort() with the only difference that it explictly specifies the timeout given for all the in-progress close operations to complete. If timeout is reached and the close operations haven't finished, then socket is forced to close.

To wait infinitely for the close operations to complete use Timeout.Infinite

Abort

```
void Abort()
```

Return type `void`

Summary

Abort this connection and all its channels.

Remarks

Note that all active channels, sessions, and models will be closed if this method is called. In comparison to normal Close() method, Abort() will not throw AlreadyClosedException or IOException during closing connection. This method waits infinitely for the in-progress close operation to complete.

Abort

```
void Abort(ushort reasonCode, string reasonText, int timeout)
```

Return type `void`

	Name	Type
Parameters	reasonCode	`ushort`
	reasonText	`string`
	timeout	`int`

Summary

Abort this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.

Remarks

The method behaves in the same way as Abort(timeout), with the only difference that the connection is closed with the given connection close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the connection

Close

```
void Close(ushort reasonCode, string reasonText)
```

Return type `void`

	Name	Type
Parameters	reasonCode	`ushort`
	reasonText	`string`

Summary

Close this connection and all its channels.

Remarks

The method behaves in the same way as Close(), with the only difference that the connection is closed with the given connection close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the connection

Close

```
void Close()
```

Return type `void`

Summary

Close this connection and all its channels.

Remarks

Note that all active channels, sessions, and models will be closed if this method is called. It will wait for the in-progress close operation to complete. This method will not return to the caller until the shutdown is complete. If the connection is already closed (or closing), then this method will throw AlreadyClosedException. It can also throw IOException when socket was closed unexpectedly.

Close

```
void Close(int timeout)
```

Return type `void`

Parameters	Name	Type
	timeout	`int`

Summary

Close this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.

Remarks

Note that all active channels, sessions, and models will be closed if this method is called. It will wait for the in-progress close operation to complete with a timeout. If the connection is already closed (or closing), then this method will throw AlreadyClosedException. It can also throw IOException when socket was closed unexpectedly. If timeout is reached and the close operations haven't finished, then socket is forced to close.

To wait infinitely for the close operations to complete use Timeout.Infinite

Close

```
void Close(ushort reasonCode, string reasonText, int timeout)
```

Return type `void`

Parameters	Name	Type
	reasonCode	`ushort`
	reasonText	`string`
	timeout	`int`

Summary

Close this connection and all its channels and wait with a timeout for all the in-progress close operations to complete.

Remarks

The method behaves in the same way as Close(int timeout), with the only difference that the connection is closed with the given connection close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the connection

CreateModel

```
IModel CreateModel()
```

Return type IModel

Summary

Create and return a fresh channel, session, and model.

Index | Namespace RabbitMQ.Client

public interface IContentHeader

- implements `ICloneable`

Summary

A decoded AMQP content header frame.

Property Summary

Type	Name	Summary
`int`	`ProtocolClassId` (r)	Retrieve the AMQP class ID of this content header.
`string`	`ProtocolClassName` (r)	Retrieve the AMQP class name of this content header.

Property Detail

int ProtocolClassId (r)

Summary

Retrieve the AMQP class ID of this content header.

string ProtocolClassName (r)

Summary

Retrieve the AMQP class name of this content header.

Index | Namespace RabbitMQ.Client

public interface IFileProperties

- implements `ICloneable`
- implements `IContentHeader`

Summary

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.

Remarks

The specification code generator provides protocol-version-specific implementations of this interface. To obtain an implementation of this interface in a protocol-version-neutral way, use IModel.CreateFileProperties().

Each property is readable, writable and clearable: a cleared property will not be transmitted over the wire. Properties on a fresh instance are clear by default.

Property Summary

Type	Name	Summary
`string`	`ClusterId` (rw)	intra-cluster routing identifier
`string`	`ContentEncoding` (rw)	MIME content encoding
`string`	`ContentType` (rw)	MIME content type
`string`	`Filename` (rw)	message filename
`IDictionary`	`Headers` (rw)	message header field table
`string`	`MessageId` (rw)	application message identifier
`byte`	`Priority` (rw)	message priority, 0 to 9
`string`	`ReplyTo` (rw)	destination to reply to
`AmqpTimestamp`	`Timestamp` (rw)	message timestamp

Method Summary

Name	Summary
`void ClearClusterId()`	Clear the ClusterId property.
`void ClearContentEncoding()`	Clear the ContentEncoding property.
`void ClearContentType()`	Clear the ContentType property.
`void ClearFilename()`	Clear the Filename property.
`void ClearHeaders()`	Clear the Headers property.
`void ClearMessageId()`	Clear the MessageId property.
`void ClearPriority()`	Clear the Priority property.
`void ClearReplyTo()`	Clear the ReplyTo property.
`void ClearTimestamp()`	Clear the Timestamp property.
`bool IsClusterIdPresent()`	Returns true iff the ClusterId property is present.
`bool IsContentEncodingPresent()`	Returns true iff the ContentEncoding property is present.
`bool IsContentTypePresent()`	Returns true iff the ContentType property is present.
`bool IsFilenamePresent()`	Returns true iff the Filename property is present.
`bool IsHeadersPresent()`	Returns true iff the Headers property is present.
`bool IsMessageIdPresent()`	Returns true iff the MessageId property is present.
`bool IsPriorityPresent()`	Returns true iff the Priority property is present.
`bool IsReplyToPresent()`	Returns true iff the ReplyTo property is present.
`bool IsTimestampPresent()`	Returns true iff the Timestamp property is present.

Property Detail

string ClusterId (rw)

Summary

intra-cluster routing identifier

string ContentEncoding (rw)

Summary

MIME content encoding

string ContentType (rw)

Summary

MIME content type

string Filename (rw)

Summary

message filename

IDictionary Headers (rw)

Summary

message header field table

string MessageId (rw)

Summary

application message identifier

byte Priority (rw)

Summary

message priority, 0 to 9

string ReplyTo (rw)

Summary

destination to reply to

AmqpTimestamp Timestamp (rw)

Summary

message timestamp

Method Detail

ClearClusterId

```
void ClearClusterId()
```

Return type `void`
Summary

Clear the ClusterId property.

ClearContentEncoding

```
void ClearContentEncoding()
```

Return type `void`
Summary

Clear the ContentEncoding property.

ClearContentType

```
void ClearContentType()
```

Return type `void`
Summary

Clear the ContentType property.

ClearFilename

```
void ClearFilename()
```

Return type `void`
Summary

Clear the Filename property.

ClearHeaders

```
void ClearHeaders()
```

Return type `void`
Summary

Clear the Headers property.

ClearMessageId

```
void ClearMessageId()
```

Return type `void`
Summary

Clear the MessageId property.

ClearPriority

```
void ClearPriority()
```

Return type `void`
Summary

Clear the Priority property.

ClearReplyTo

```
void ClearReplyTo()
```

Return type `void`
Summary

Clear the ReplyTo property.

ClearTimestamp

```
void ClearTimestamp()
```

Return type `void`
Summary

Clear the Timestamp property.

IsClusterIdPresent

```
bool IsClusterIdPresent()
```

Return type `bool`
Summary

Returns true iff the ClusterId property is present.

IsContentEncodingPresent

```
bool IsContentEncodingPresent()
```

Return type `bool`
Summary

Returns true iff the ContentEncoding property is present.

IsContentTypePresent

```
bool IsContentTypePresent()
```

Return type `bool`
Summary

Returns true iff the ContentType property is present.

IsFilenamePresent

```
bool IsFilenamePresent()
```

Return type `bool`

Summary

Returns true iff the Filename property is present.

IsHeadersPresent

```
bool IsHeadersPresent()
```

Return type bool
Summary

Returns true iff the Headers property is present.

IsMessageIdPresent

```
bool IsMessageIdPresent()
```

Return type bool
Summary

Returns true iff the MessageId property is present.

IsPriorityPresent

```
bool IsPriorityPresent()
```

Return type bool
Summary

Returns true iff the Priority property is present.

IsReplyToPresent

```
bool IsReplyToPresent()
```

Return type bool
Summary

Returns true iff the ReplyTo property is present.

IsTimestampPresent

```
bool IsTimestampPresent()
```

Return type bool
Summary

Returns true iff the Timestamp property is present.
Index | Namespace RabbitMQ.Client

public interface IMethod

Summary

A decoded AMQP method frame.

Remarks

AMQP methods can be RPC requests, RPC responses, exceptions (ChannelClose, ConnectionClose), or one-way asynchronous messages. Currently this information is not recorded in their type or interface: it is implicit in the way the method is used, and the way it is defined in the AMQP specification. A future revision of the RabbitMQ .NET client library may extend the IMethod interface to represent this information explicitly.

Property Summary

Type	Name	Summary
`int`	`ProtocolClassId` (r)	Retrieves the class ID number of this method, as defined in the AMQP specification XML.
`int`	`ProtocolMethodId` (r)	Retrieves the method ID number of this method, as defined in the AMQP specification XML.
`string`	`ProtocolMethodName` (r)	Retrieves the name of this method - for debugging use.

Property Detail

int ProtocolClassId (r)

Summary

Retrieves the class ID number of this method, as defined in the AMQP specification XML.

int ProtocolMethodId (r)

Summary

Retrieves the method ID number of this method, as defined in the AMQP specification XML.

string ProtocolMethodName (r)

Summary

Retrieves the name of this method - for debugging use.

Index | Namespace RabbitMQ.Client

public interface IModel

- implements `IDisposable`

Summary

Common AMQP model, spanning the union of the functionality offered by versions 0-8, 0-8qpid, 0-9 and 0-9-1 of AMQP.

Remarks

Extends the IDisposable interface, so that the "using" statement can be used to scope the lifetime of a channel when appropriate.

Property Summary

Type	Name	Summary
ShutdownEventArgs	CloseReason (r)	Returns null if the session is still in a state where it can be used, or the cause of its closure otherwise.
IBasicConsumer	DefaultConsumer (rw)	Signalled when an unexpected message is delivered Under certain circumstances it is possible for a channel to receive a message delivery which does not match any consumer which is currently set up via basicConsume(). This will occur after the following sequence of events: ctag = basicConsume(queue, consumer); // i.e. with explicit acks // some deliveries take place but are not acked basicCancel(ctag); basicRecover(false); Since requeue is specified to be false in the basicRecover, the spec states that the message must be redelivered to "the original recipient" - i.e. the same channel / consumer-tag. But the consumer is no longer active. In these circumstances, you can register a default consumer to handle such deliveries. If no default consumer is registered an InvalidOperationException will be thrown when such a delivery arrives. Most people will not need to use this.
bool	IsOpen (r)	Returns true if the session is still in a state where it can be used. Identical to checking if CloseReason == null.
ulong	NextPublishSeqNo (r)	When in confirm mode, return the sequence number of the next message to be published.

Event Summary

Type	Name	Summary
BasicAckEventHandler	BasicAcks	Signalled when a Basic.Ack command arrives from the broker.
BasicNackEventHandler	BasicNacks	Signalled when a Basic.Nack command arrives from the broker.
BasicRecoverOkEventHandler	BasicRecoverOk	All messages received before this fires that haven't been ack'ed will be redelivered. All messages received afterwards won't be. Handlers for this event are invoked by the connection thread. It is sometimes useful to allow that thread to know that a recover-ok has been received, rather than the thread that invoked BasicRecover().
BasicReturnEventHandler	BasicReturn	Signalled when a Basic.Return command arrives from the broker.
CallbackExceptionEventHandler	CallbackException	Signalled when an exception occurs in a callback invoked by the model.
FlowControlEventHandler	FlowControl	(undocumented)
ModelShutdownEventHandler	ModelShutdown	Notifies the destruction of the model.

Method Summary

Name	Summary
void Abort(ushort replyCode, string replyText)	Abort this session.
void Abort()	Abort this session.
void BasicAck(ulong deliveryTag, bool multiple)	(Spec method) Acknowledge one or more delivered message(s).
void BasicCancel(string consumerTag)	Delete a Basic content-class consumer.
string BasicConsume(string queue, bool noAck, string consumerTag, IBasicConsumer consumer)	Start a Basic content-class consumer.
string BasicConsume(string queue, bool noAck, IBasicConsumer consumer)	Start a Basic content-class consumer.
string BasicConsume(string queue, bool noAck, string consumerTag, bool noLocal, bool exclusive, IDictionary arguments, IBasicConsumer consumer)	Start a Basic content-class consumer.
string BasicConsume(string queue, bool noAck, string consumerTag, IDictionary arguments, IBasicConsumer consumer)	Start a Basic content-class consumer.
BasicGetResult BasicGet(string queue, bool noAck)	(Spec method) Retrieve an individual message, if one is available; returns null if the server answers that no messages are currently available. See also IModel.BasicAck.
void BasicNack(ulong deliveryTag, bool multiple, bool requeue)	Reject one or more delivered message(s).
void BasicPublish(string exchange, string routingKey, bool mandatory, IBasicProperties basicProperties, byte[] body)	(Spec method) Convenience overload of BasicPublish.
void BasicPublish(string exchange, string routingKey, bool mandatory, bool immediate, IBasicProperties basicProperties, byte[] body)	(Spec method) Publish a message using the Basic content-class.
void BasicPublish(PublicationAddress addr, IBasicProperties basicProperties, byte[] body)	(Spec method) Convenience overload of BasicPublish.
void BasicPublish(string exchange, string routingKey, IBasicProperties basicProperties, byte[] body)	(Spec method) Convenience overload of BasicPublish.
void BasicQos(uint prefetchSize, ushort prefetchCount, bool global)	(Spec method) Configures QoS parameters of the Basic content-class.
void BasicRecover(bool requeue)	(Spec method)
void BasicRecoverAsync(bool requeue)	(Spec method)
void BasicReject(ulong deliveryTag, bool requeue)	(Spec method) Reject a delivered message.
void ChannelFlow(bool active)	(Spec method) Channel flow control functionality.
void Close()	Close this session.
void Close(ushort replyCode, string replyText)	Close this session.
void ConfirmSelect()	Enable publisher acknowledgements.
IBasicProperties CreateBasicProperties()	Construct a completely empty content header for use with the Basic content class.
IFileProperties CreateFileProperties()	Construct a completely empty content header for use with the File content class. (unsupported in AMQP 0-9-1)
IStreamProperties CreateStreamProperties()	Construct a completely empty content header for use with the Stream content class. (unsupported in AMQP 0-9-1)

void DtxSelect()	(Spec method) Enable DTX mode for this session. (unsupported in AMQP 0-9-1)
void DtxStart(string dtxIdentifier)	(Spec method, unsupported in AMQP 0-9-1)
void ExchangeBind(string destination, string source, string routingKey, IDictionary arguments)	(Extension method) Bind an exchange to an exchange.
void ExchangeBind(string destination, string source, string routingKey)	(Extension method) Bind an exchange to an exchange.
void ExchangeDeclare(string exchange, string type)	(Spec method) Declare an exchange.
void ExchangeDeclare(string exchange, string type, bool durable)	(Spec method) Declare an exchange.
void ExchangeDeclare(string exchange, string type, bool durable, bool autoDelete, IDictionary arguments)	(Spec method) Declare an exchange.
void ExchangeDeclarePassive(string exchange)	(Spec method) Declare an exchange.
void ExchangeDelete(string exchange, bool ifUnused)	(Spec method) Delete an exchange.
void ExchangeDelete(string exchange)	(Spec method) Delete an exchange.
void ExchangeUnbind(string destination, string source, string routingKey)	(Extension method) Unbind an exchange from an exchange.
void ExchangeUnbind(string destination, string source, string routingKey, IDictionary arguments)	(Extension method) Unbind an exchange from an exchange.
void QueueBind(string queue, string exchange, string routingKey, IDictionary arguments)	(Spec method) Bind a queue to an exchange.
void QueueBind(string queue, string exchange, string routingKey)	(Spec method) Bind a queue to an exchange.
QueueDeclareOk QueueDeclare(string queue, bool durable, bool exclusive, bool autoDelete, IDictionary arguments)	(Spec method) Declare a queue.
QueueDeclareOk QueueDeclare()	(Spec method) Declare a queue.
QueueDeclareOk QueueDeclarePassive(string queue)	Declare a queue passively.
uint QueueDelete(string queue)	(Spec method) Delete a queue.
uint QueueDelete(string queue, bool ifUnused, bool ifEmpty)	(Spec method) Delete a queue.
uint QueuePurge(string queue)	(Spec method) Purge a queue of messages.
void QueueUnbind(string queue, string exchange, string routingKey, IDictionary arguments)	(Spec method) Unbind a queue from an exchange.
void TxCommit()	(Spec method) Commit this session's active TX transaction.
void TxRollback()	(Spec method) Roll back this session's active TX transaction.
void TxSelect()	(Spec method) Enable TX mode for this session.
bool WaitForConfirms(TimeSpan timeout, out bool timedOut)	Wait until all published messages have been confirmed.
bool WaitForConfirms()	Wait until all published messages have been confirmed.
void WaitForConfirmsOrDie()	Wait until all published messages have been confirmed.
void WaitForConfirmsOrDie(TimeSpan timeout)	Wait until all published messages have been confirmed.

Property Detail

ShutdownEventArgs CloseReason (r)

Summary

Returns null if the session is still in a state where it can be used, or the cause of its closure otherwise.

IBasicConsumer DefaultConsumer (rw)

Summary

Signalled when an unexpected message is delivered Under certain circumstances it is possible for a channel to receive a message delivery which does not match any consumer which is currently set up via basicConsume(). This will occur after the following sequence of events: ctag = basicConsume(queue, consumer); // i.e. with explicit acks // some deliveries take place but are not acked basicCancel(ctag); basicRecover(false); Since requeue is specified to be false in the basicRecover, the spec states that the message must be redelivered to "the original recipient" - i.e. the same channel / consumer-tag. But the consumer is no longer active. In these circumstances, you can register a default consumer to handle such deliveries. If no default consumer is registered an InvalidOperationException will be thrown when such a delivery arrives. Most people will not need to use this.

bool IsOpen (r)

Summary

Returns true if the session is still in a state where it can be used. Identical to checking if CloseReason == null.

ulong NextPublishSeqNo (r)

Summary

When in confirm mode, return the sequence number of the next message to be published.

Event Detail

BasicAckEventHandler BasicAcks

Summary

Signalled when a Basic.Ack command arrives from the broker.

BasicNackEventHandler BasicNacks

Summary

Signalled when a Basic.Nack command arrives from the broker.

BasicRecoverOkEventHandler BasicRecoverOk

Summary

All messages received before this fires that haven't been ack'ed will be redelivered. All messages received afterwards won't be. Handlers for this event are invoked by the connection thread. It is sometimes useful to allow that thread to know that a recover-ok has been received, rather than the thread that invoked BasicRecover().

BasicReturnEventHandler BasicReturn

Summary

Signalled when a Basic.Return command arrives from the broker.

CallbackExceptionEventHandler CallbackException

Summary

Signalled when an exception occurs in a callback invoked by the model.

Remarks

Examples of cases where this event will be signalled include exceptions thrown in IBasicConsumer methods, or exceptions thrown in ModelShutdownEventHandler delegates etc.

FlowControlEventHandler FlowControl

ModelShutdownEventHandler ModelShutdown

Summary

Notifies the destruction of the model.

Remarks

If the model is already destroyed at the time an event handler is added to this event, the event handler will be fired immediately.

Method Detail

Abort

```
void Abort(ushort replyCode, string replyText)
```

Return type `void`

	Name	Type
Parameters	replyCode	`ushort`
	replyText	`string`

Summary

Abort this session.

Remarks

The method behaves in the same way as Abort(), with the only difference that the model is closed with the given model close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the model

Abort

```
void Abort()
```

Return type `void`

Summary

Abort this session.

Remarks

If the session is already closed (or closing), then this method does nothing but wait for the in-progress close operation to complete. This method will not return to the caller until the shutdown is complete. In comparison to normal Close() method, Abort() will not throw AlreadyClosedException or IOException during closing model.

BasicAck

```
void BasicAck(ulong deliveryTag, bool multiple)
```

Return type `void`

	Name	Type
Parameters	deliveryTag	`ulong`
	multiple	`bool`

Summary

(Spec method) Acknowledge one or more delivered message(s).

BasicCancel

```
void BasicCancel(string consumerTag)
```

Return type `void`

	Name	Type
Parameters	consumerTag	`string`

Summary

Delete a Basic content-class consumer.

BasicConsume

```
string BasicConsume(string queue, bool noAck, string consumerTag, IBasicConsumer
consumer)
```

Return type `string`

	Name	Type
Parameters	queue	`string`
	noAck	`bool`
	consumerTag	`string`
	consumer	`IBasicConsumer`

Summary

Start a Basic content-class consumer.

Remarks

The consumer is started with an empty consumer tag (i.e. the server creates and returns a fresh consumer tag), noLocal=false and exclusive=false.

BasicConsume

```
string BasicConsume(string queue, bool noAck, IBasicConsumer consumer)
```

Return type `string`

	Name	Type
Parameters	queue	`string`
	noAck	`bool`
	consumer	`IBasicConsumer`

Summary

Start a Basic content-class consumer.

Remarks

The consumer is started with noAck=false (i.e. BasicAck is required), an empty consumer tag (i.e. the server creates and returns a fresh consumer tag), noLocal=false and exclusive=false.

BasicConsume

```
string BasicConsume(string queue, bool noAck, string consumerTag, bool noLocal, bool
exclusive, IDictionary arguments, IBasicConsumer consumer)
```

Return type `string`

	Name	Type
Parameters	queue	`string`
	noAck	`bool`
	consumerTag	`string`
	noLocal	`bool`
	exclusive	`bool`
	arguments	`IDictionary`
	consumer	`IBasicConsumer`

Summary

Start a Basic content-class consumer.

BasicConsume

```
string BasicConsume(string queue, bool noAck, string consumerTag, IDictionary arguments,
IBasicConsumer consumer)
```

Return type `string`

	Name	Type
Parameters	queue	`string`
	noAck	`bool`
	consumerTag	`string`
	arguments	`IDictionary`
	consumer	`IBasicConsumer`

Summary

Start a Basic content-class consumer.

Remarks

The consumer is started with noLocal=false and exclusive=false.

BasicGet

```
BasicGetResult BasicGet(string queue, bool noAck)
```

Return type `BasicGetResult`

	Name	Type
Parameters	queue	`string`
	noAck	`bool`

Summary

(Spec method) Retrieve an individual message, if one is available; returns null if the server answers that no messages are currently available. See also IModel.BasicAck.

BasicNack

```
void BasicNack(ulong deliveryTag, bool multiple, bool requeue)
```

Return type `void`

	Name	Type
Parameters	deliveryTag	`ulong`
	multiple	`bool`
	requeue	`bool`

Summary

Reject one or more delivered message(s).

BasicPublish

```
void BasicPublish(string exchange, string routingKey, bool mandatory, IBasicProperties
basicProperties, byte[] body)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	routingKey	`string`
	mandatory	`bool`
	basicProperties	`IBasicProperties`
	body	`byte[]`

Summary

(Spec method) Convenience overload of BasicPublish.

Remarks

The publication occurs with immediate=false.

BasicPublish

```
void BasicPublish(string exchange, string routingKey, bool mandatory, bool immediate,
IBasicProperties basicProperties, byte[] body)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	routingKey	`string`
	mandatory	`bool`
	immediate	`bool`
	basicProperties	`IBasicProperties`
	body	`byte[]`

Summary

(Spec method) Publish a message using the Basic content-class.

Remarks

Note that the RabbitMQ server does not support the 'immediate' flag.

BasicPublish

```
void BasicPublish(PublicationAddress addr, IBasicProperties basicProperties, byte[] body)
```

Return type `void`

	Name	Type
Parameters	addr	PublicationAddress
	basicProperties	IBasicProperties
	body	byte[]

Summary

(Spec method) Convenience overload of BasicPublish.

Remarks

The publication occurs with mandatory=false and immediate=false.

BasicPublish

```
void BasicPublish(string exchange, string routingKey, IBasicProperties basicProperties,
byte[] body)
```

Return type `void`

	Name	Type
Parameters	exchange	string
	routingKey	string
	basicProperties	IBasicProperties
	body	byte[]

Summary

(Spec method) Convenience overload of BasicPublish.

Remarks

The publication occurs with mandatory=false and immediate=false.

BasicQos

```
void BasicQos(uint prefetchSize, ushort prefetchCount, bool global)
```

Return type `void`

	Name	Type
Parameters	prefetchSize	uint
	prefetchCount	ushort
	global	bool

Summary

(Spec method) Configures QoS parameters of the Basic content-class.

BasicRecover

```
void BasicRecover(bool requeue)
```

Return type `void`

Parameters

Name	Type
requeue	`bool`

Summary

(Spec method)

BasicRecoverAsync

```
void BasicRecoverAsync(bool requeue)
```

Return type `void`

Parameters

Name	Type
requeue	`bool`

Summary

(Spec method)

BasicReject

```
void BasicReject(ulong deliveryTag, bool requeue)
```

Return type `void`

Parameters

Name	Type
deliveryTag	`ulong`
requeue	`bool`

Summary

(Spec method) Reject a delivered message.

ChannelFlow

```
void ChannelFlow(bool active)
```

Return type `void`

Parameters

Name	Type
active	`bool`

Summary

(Spec method) Channel flow control functionality.

Remarks

Close

```
void Close()
```

Return type `void`

Summary

Close this session.

Remarks

If the session is already closed (or closing), then this method does nothing but wait for the in-progress close operation to complete. This method will not return to the caller until the shutdown is complete.

Close

```
void Close(ushort replyCode, string replyText)
```

Return type `void`

	Name	Type
Parameters	replyCode	`ushort`
	replyText	`string`

Summary

Close this session.

Remarks

The method behaves in the same way as Close(), with the only difference that the model is closed with the given model close code and message.

The close code (See under "Reply Codes" in the AMQP specification)

A message indicating the reason for closing the model

ConfirmSelect

```
void ConfirmSelect()
```

Return type `void`

Summary

Enable publisher acknowledgements.

CreateBasicProperties

```
IBasicProperties CreateBasicProperties()
```

Return type `IBasicProperties`

Summary

Construct a completely empty content header for use with the Basic content class.

CreateFileProperties

```
IFileProperties CreateFileProperties()
```

Return type `IFileProperties`

Summary

Construct a completely empty content header for use with the File content class. (unsupported in AMQP 0-9-1)

CreateStreamProperties

```
IStreamProperties CreateStreamProperties()
```

Return type IStreamProperties

Summary

Construct a completely empty content header for use with the Stream content class. (unsupported in AMQP 0-9-1)

DtxSelect

```
void DtxSelect()
```

Return type void

Summary

(Spec method) Enable DTX mode for this session. (unsupported in AMQP 0-9-1)

DtxStart

```
void DtxStart(string dtxIdentifier)
```

Return type void

Parameters

Name	Type
dtxIdentifier	string

Summary

(Spec method, unsupported in AMQP 0-9-1)

ExchangeBind

```
void ExchangeBind(string destination, string source, string routingKey, IDictionary
arguments)
```

Return type void

Parameters

Name	Type
destination	string
source	string
routingKey	string
arguments	IDictionary

Summary

(Extension method) Bind an exchange to an exchange.

ExchangeBind

```
void ExchangeBind(string destination, string source, string routingKey)
```

Return type void

Parameters

Name	Type
destination	string
source	string
routingKey	string

Summary

(Extension method) Bind an exchange to an exchange.

ExchangeDeclare

```
void ExchangeDeclare(string exchange, string type)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	type	`string`

Summary

(Spec method) Declare an exchange.

Remarks

The exchange is declared non-passive, non-durable, non-autodelete, and non-internal, with no arguments. The "nowait" option is not exercised.

ExchangeDeclare

```
void ExchangeDeclare(string exchange, string type, bool durable)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	type	`string`
	durable	`bool`

Summary

(Spec method) Declare an exchange.

Remarks

The exchange is declared non-passive, non-autodelete, and non-internal, with no arguments. The "nowait" option is not exercised.

ExchangeDeclare

```
void ExchangeDeclare(string exchange, string type, bool durable, bool autoDelete,
IDictionary arguments)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	type	`string`
	durable	`bool`
	autoDelete	`bool`
	arguments	`IDictionary`

Summary

(Spec method) Declare an exchange.

Remarks

The exchange is declared non-passive and non-internal. The "nowait" option is not exercised.

ExchangeDeclarePassive

```
void ExchangeDeclarePassive(string exchange)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`

Summary

(Spec method) Declare an exchange.

Remarks

The exchange is declared passive.

ExchangeDelete

```
void ExchangeDelete(string exchange, bool ifUnused)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`
	ifUnused	`bool`

Summary

(Spec method) Delete an exchange.

ExchangeDelete

```
void ExchangeDelete(string exchange)
```

Return type `void`

	Name	Type
Parameters	exchange	`string`

Summary

(Spec method) Delete an exchange.

Remarks

The exchange is deleted regardless of any queue bindings.

ExchangeUnbind

```
void ExchangeUnbind(string destination, string source, string routingKey)
```

Return type `void`

	Name	Type
Parameters	destination	`string`
	source	`string`
	routingKey	`string`

Summary

(Extension method) Unbind an exchange from an exchange.

ExchangeUnbind

```
void ExchangeUnbind(string destination, string source, string routingKey, IDictionary
arguments)
```

Return type `void`

	Name	Type
Parameters	destination	`string`
	source	`string`
	routingKey	`string`
	arguments	`IDictionary`

Summary

(Extension method) Unbind an exchange from an exchange.

QueueBind

```
void QueueBind(string queue, string exchange, string routingKey, IDictionary arguments)
```

Return type `void`

	Name	Type
Parameters	queue	`string`
	exchange	`string`
	routingKey	`string`
	arguments	`IDictionary`

Summary

(Spec method) Bind a queue to an exchange.

QueueBind

```
void QueueBind(string queue, string exchange, string routingKey)
```

Return type `void`

	Name	Type
Parameters	queue	`string`
	exchange	`string`
	routingKey	`string`

Summary

(Spec method) Bind a queue to an exchange.

QueueDeclare

```
QueueDeclareOk QueueDeclare(string queue, bool durable, bool exclusive, bool autoDelete,
IDictionary arguments)
```

Return type `QueueDeclareOk`

	Name	Type
Parameters	queue	`string`
	durable	`bool`
	exclusive	`bool`
	autoDelete	`bool`
	arguments	`IDictionary`

Summary

(Spec method) Declare a queue.

QueueDeclare

```
QueueDeclareOk QueueDeclare()
```

Return type QueueDeclareOk

Summary

(Spec method) Declare a queue.

Remarks

The queue is declared non-passive, non-durable, but exclusive and autodelete, with no arguments. The server autogenerates a name for the queue - the generated name is the return value of this method.

QueueDeclarePassive

```
QueueDeclareOk QueueDeclarePassive(string queue)
```

Return type QueueDeclareOk

	Name	Type
Parameters	queue	`string`

Summary

Declare a queue passively.

Remarks

The queue is declared passive, non-durable, non-exclusive, and non-autodelete, with no arguments. The queue is declared passively; i.e. only check if it exists.

QueueDelete

```
uint QueueDelete(string queue)
```

Return type `uint`

	Name	Type
Parameters	queue	`string`

Summary

(Spec method) Delete a queue.

Remarks

Returns the number of messages purged during queue deletion.

QueueDelete

```
uint QueueDelete(string queue, bool ifUnused, bool ifEmpty)
```

Return type `uint`

	Name	Type
Parameters	queue	`string`
	ifUnused	`bool`
	ifEmpty	`bool`

Summary

(Spec method) Delete a queue.

Remarks

Returns the number of messages purged during queue deletion.

`uint.MaxValue`

.

QueuePurge

```
uint QueuePurge(string queue)
```

Return type `uint`

	Name	Type
Parameters	queue	`string`

Summary

(Spec method) Purge a queue of messages.

Remarks

Returns the number of messages purged.

QueueUnbind

```
void QueueUnbind(string queue, string exchange, string routingKey, IDictionary arguments)
```

Return type `void`

	Name	Type
Parameters	queue	`string`
	exchange	`string`
	routingKey	`string`
	arguments	`IDictionary`

Summary

(Spec method) Unbind a queue from an exchange.

Remarks

Note: This operation is only supported when communicating using AMQP protocol version 0-9, or when communicating with a 0-8 broker that has been enhanced with the unofficial addition of a queue.unbind method.

TxCommit

```
void TxCommit()
```

Return type `void`

Summary

(Spec method) Commit this session's active TX transaction.

TxRollback

```
void TxRollback()
```

Return type `void`
Summary

(Spec method) Roll back this session's active TX transaction.

TxSelect

```
void TxSelect()
```

Return type `void`
Summary

(Spec method) Enable TX mode for this session.

WaitForConfirms

```
bool WaitForConfirms(TimeSpan timeout, out bool timedOut)
```

Return type `bool`

	Name	Type
Parameters	timeout	`TimeSpan`
	timedOut	`out bool`

Summary

Wait until all published messages have been confirmed.
Returns

true if no nacks were received within the timeout, otherwise false
Param

How long to wait (at most) before returning whether or not any nacks were returned
Param

True if the method returned because the timeout elapsed, not because all messages were ack'd or at least one nack'd.
Remarks

Waits until all messages published since the last call have been either ack'd or nack'd by the broker. Returns whether all the messages were ack'd (and none were nack'd). Note, throws an exception when called on a non-Confirm channel.

WaitForConfirms

```
bool WaitForConfirms()
```

Return type `bool`
Summary

Wait until all published messages have been confirmed.

Remarks

Waits until all messages published since the last call have been either ack'd or nack'd by the broker. Returns whether all the messages were ack'd (and none were nack'd). Note, throws an exception when called on a non-Confirm channel.

WaitForConfirmsOrDie

```
void WaitForConfirmsOrDie()
```

Return type `void`

Summary

Wait until all published messages have been confirmed.

Remarks

Waits until all messages published since the last call have been ack'd by the broker. If a nack is received, throws an OperationInterrupedException exception immediately.

WaitForConfirmsOrDie

```
void WaitForConfirmsOrDie(TimeSpan timeout)
```

Return type `void`

Parameters	Name	Type
	timeout	`TimeSpan`

Summary

Wait until all published messages have been confirmed.

Remarks

Waits until all messages published since the last call have been ack'd by the broker. If a nack is received or the timeout elapses, throws an OperationInterrupedException exception immediately.

Index | Namespace RabbitMQ.Client

public interface IProtocol

Summary

Object describing various overarching parameters associated with a particular AMQP protocol variant.

Property Summary

Type	Name	Summary
`string`	`ApiName` (r)	Retrieve the protocol's API name, used for printing, configuration properties, IDE integration, Protocols.cs etc.
`int`	`DefaultPort` (r)	Retrieve the protocol's default TCP port
`int`	`MajorVersion` (r)	Retrieve the protocol's major version number
`int`	`MinorVersion` (r)	Retrieve the protocol's minor version number
`int`	`Revision` (r)	Retrieve the protocol's revision (if specified)

Method Summary

Name	Summary
`IConnection CreateConnection(ConnectionFactory factory, bool insist, IFrameHandler frameHandler)`	Construct a connection from a given set of parameters and a frame handler. The "insist" parameter is passed on to the AMQP connection.open method.
`IFrameHandler CreateFrameHandler(AmqpTcpEndpoint endpoint, ConnectionFactory.ObtainSocket socketFactory, int timeout)`	Construct a frame handler for a given endpoint.
`IModel CreateModel(ISession session)`	Construct a protocol model atop a given session.

Property Detail

string ApiName (r)

Summary

Retrieve the protocol's API name, used for printing, configuration properties, IDE integration, Protocols.cs etc.

int DefaultPort (r)

Summary

Retrieve the protocol's default TCP port

int MajorVersion (r)

Summary

Retrieve the protocol's major version number

int MinorVersion (r)

Summary

Retrieve the protocol's minor version number

int Revision (r)

Summary

Retrieve the protocol's revision (if specified)

Method Detail

CreateConnection

```
IConnection CreateConnection(ConnectionFactory factory, bool insist, IFrameHandler
frameHandler)
```

Return type IConnection

	Name	Type
Parameters	factory	ConnectionFactory
	insist	bool
	frameHandler	IFrameHandler

Summary

Construct a connection from a given set of parameters and a frame handler. The "insist" parameter is passed on to the AMQP connection.open method.

CreateFrameHandler

```
IFrameHandler CreateFrameHandler(AmqpTcpEndpoint endpoint, ConnectionFactory.ObtainSocket
socketFactory, int timeout)
```

Return type IFrameHandler

	Name	Type
Parameters	endpoint	AmqpTcpEndpoint
	socketFactory	ConnectionFactory.ObtainSocket
	timeout	int

Summary

Construct a frame handler for a given endpoint.

CreateModel

```
IModel CreateModel(ISession session)
```

Return type IModel

	Name	Type
Parameters	session	ISession

Summary

Construct a protocol model atop a given session.

Index | Namespace RabbitMQ.Client

public interface IStreamProperties

- implements ICloneable
- implements IContentHeader

Summary

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.

Remarks

The specification code generator provides protocol-version-specific implementations of this interface. To obtain an implementation of this interface in a protocol-version-neutral way, use IModel.CreateStreamProperties().

Each property is readable, writable and clearable: a cleared property will not be transmitted over the wire. Properties on a fresh instance are clear by default.

Property Summary

Type	Name	Summary
string	ContentEncoding (rw)	MIME content encoding
string	ContentType (rw)	MIME content type
IDictionary	Headers (rw)	message header field table
byte	Priority (rw)	message priority, 0 to 9
AmqpTimestamp	Timestamp (rw)	message timestamp

Method Summary

Name	Summary
void ClearContentEncoding()	Clear the ContentEncoding property.
void ClearContentType()	Clear the ContentType property.
void ClearHeaders()	Clear the Headers property.
void ClearPriority()	Clear the Priority property.
void ClearTimestamp()	Clear the Timestamp property.
bool IsContentEncodingPresent()	Returns true iff the ContentEncoding property is present.
bool IsContentTypePresent()	Returns true iff the ContentType property is present.
bool IsHeadersPresent()	Returns true iff the Headers property is present.
bool IsPriorityPresent()	Returns true iff the Priority property is present.
bool IsTimestampPresent()	Returns true iff the Timestamp property is present.

Property Detail

string ContentEncoding (rw)

Summary

MIME content encoding

string ContentType (rw)

Summary

MIME content type

IDictionary Headers (rw)

Summary

message header field table

byte Priority (rw)

Summary

message priority, 0 to 9

AmqpTimestamp Timestamp (rw)

Summary

message timestamp

Method Detail

ClearContentEncoding

```
void ClearContentEncoding()
```

Return type `void`

Summary

Clear the ContentEncoding property.

ClearContentType

```
void ClearContentType()
```

Return type `void`

Summary

Clear the ContentType property.

ClearHeaders

```
void ClearHeaders()
```

Return type `void`

Summary

Clear the Headers property.

ClearPriority

```
void ClearPriority()
```

Return type `void`

Summary

Clear the Priority property.

ClearTimestamp

```
void ClearTimestamp()
```

Return type void
Summary

Clear the Timestamp property.

IsContentEncodingPresent

```
bool IsContentEncodingPresent()
```

Return type bool
Summary

Returns true iff the ContentEncoding property is present.

IsContentTypePresent

```
bool IsContentTypePresent()
```

Return type bool
Summary

Returns true iff the ContentType property is present.

IsHeadersPresent

```
bool IsHeadersPresent()
```

Return type bool
Summary

Returns true iff the Headers property is present.

IsPriorityPresent

```
bool IsPriorityPresent()
```

Return type bool
Summary

Returns true iff the Priority property is present.

IsTimestampPresent

```
bool IsTimestampPresent()
```

Return type bool
Summary

Returns true iff the Timestamp property is present.
Index | Namespace RabbitMQ.Client

delegate ObtainSocket

- declared within ConnectionFactory
- extends MulticastDelegate

```
delegate TcpClient ConnectionFactory.ObtainSocket(AddressFamily addressFamily)
```

Return type TcpClient

Parameters

Name	Type
addressFamily	AddressFamily

Index | Namespace RabbitMQ.Client

public class ConnectionFactory

Nested types: ObtainSocket

Summary

Main entry point to the RabbitMQ .NET AMQP client API. Constructs IConnection instances.

Remarks

A simple example of connecting to a broker:

```
ConnectionFactory factory = new ConnectionFactory();
//
// The next six lines are optional:
factory.UserName = ConnectionFactory.DefaultUser;
factory.Password = ConnectionFactory.DefaultPass;
factory.VirtualHost = ConnectionFactory.DefaultVHost;
factory.Protocol = Protocols.FromEnvironment();
factory.HostName = hostName;
factory.Port     = AmqpTcpEndpoint.UseDefaultPort;
//
IConnection conn = factory.CreateConnection();
//
IModel ch = conn.CreateModel();
//
// ... use ch's IModel methods ...
//
ch.Close(Constants.ReplySuccess, "Closing the channel");
conn.Close(Constants.ReplySuccess, "Closing the connection");
```

The same example, written more compactly with AMQP URIs:

```
ConnectionFactory factory = new ConnectionFactory();
factory.Uri = "amqp://localhost";
IConnection conn = factory.CreateConnection();
...
```

Please see also the API overview and tutorial in the User Guide.

Note that the Uri property takes a string representation of an AMQP URI. Omitted URI parts will take default values. The host part of the URI cannot be omitted and URIs of the form "amqp://foo/" (note the trailling slash) also represent the default virtual host. The latter issue means that virtual hosts with an empty name are not addressable.

Field Summary

Flags	Type	Name	Summary
public	AuthMechanismFactory[]	AuthMechanisms	SASL auth mechanisms to use.
public	IDictionary	ClientProperties	Dictionary of client properties to be sent to the server
public static	AuthMechanismFactory[]	DefaultAuthMechanisms	Default SASL auth mechanisms to use.
public const	ushort	DefaultChannelMax	Default value for the desired maximum channel number, with zero meaning unlimited (value: 0)
public const	int	DefaultConnectionTimeout	Default value for connection attempt timeout, in milliseconds
public const	uint	DefaultFrameMax	Default value for the desired maximum frame size, with zero meaning unlimited (value: 0)
	ushort	DefaultHeartbeat	

public const			Default value for desired heartbeat interval, in seconds, with zero meaning none (value: 0)
public const	string	DefaultPass	Default password (value: "guest")
public const	string	DefaultUser	Default user name (value: "guest")
public const	string	DefaultVHost	Default virtual host (value: "/")
public	string	HostName	The host to connect to
public	string	Password	Password to use when authenticating to the server
public	int	Port	The port to connect on. AmqpTcpEndpoint.UseDefaultPort indicates the default for the protocol should be used.
public	IProtocol	Protocol	The AMQP protocol to be used
public	ushort	RequestedChannelMax	Maximum channel number to ask for
public	int	RequestedConnectionTimeout	Timeout setting for connection attempts (in milliseconds)
public	uint	RequestedFrameMax	Frame-max parameter to ask for (in bytes)
public	ushort	RequestedHeartbeat	Heartbeat setting to request (in seconds)
public	ConnectionFactory.ObtainSocket	SocketFactory	Set custom socket options by providing a SocketFactory
public	SslOption	Ssl	Ssl options setting
public	string	UserName	Username to use when authenticating to the server
public	string	VirtualHost	Virtual host to access during this connection

Property Summary

Flags	Type	Name	Summary
public	AmqpTcpEndpoint	Endpoint (rw)	The AMQP connection target
public	Uri	uri (w)	Set connection parameters using the amqp or amqps scheme
public	string	Uri (w)	Set connection parameters using the amqp or amqps scheme

Constructor Summary

Flags	Name	Summary
public	ConnectionFactory()	Construct a fresh instance, with all fields set to their respective defaults.

Method Summary

Flags	Name	Summary
public	AuthMechanismFactory AuthMechanismFactory(string[] mechs)	Given a list of mechanism names supported by the server, select a preferred mechanism, or null if we have none in common.
public virtual	IConnection CreateConnection()	Create a connection to the specified endpoint No broker-originated redirects are permitted.
public virtual	IConnection CreateConnection(int maxRedirects)	Create a connection to the first available endpoint in the list provided. Up to a maximum of maxRedirects broker-originated redirects are permitted for each endpoint tried.

public static	TcpClient DefaultSocketFactory(AddressFamily addressFamily) (undocumented)

Field Detail

public AuthMechanismFactory[] AuthMechanisms

Summary

SASL auth mechanisms to use.

public IDictionary ClientProperties

Summary

Dictionary of client properties to be sent to the server

public static AuthMechanismFactory[] DefaultAuthMechanisms

Summary

Default SASL auth mechanisms to use.

public const ushort DefaultChannelMax

Summary

Default value for the desired maximum channel number, with zero meaning unlimited (value: 0)

public const int DefaultConnectionTimeout

Summary

Default value for connection attempt timeout, in milliseconds

public const uint DefaultFrameMax

Summary

Default value for the desired maximum frame size, with zero meaning unlimited (value: 0)

public const ushort DefaultHeartbeat

Summary

Default value for desired heartbeat interval, in seconds, with zero meaning none (value: 0)

public const string DefaultPass

Summary

Default password (value: "guest")

public const string DefaultUser

Summary

Default user name (value: "guest")

public const string DefaultVHost

Summary

Default virtual host (value: "/")

public string HostName

Summary

The host to connect to

public string Password

Summary

Password to use when authenticating to the server

public int Port

Summary

The port to connect on. AmqpTcpEndpoint.UseDefaultPort indicates the default for the protocol should be used.

public IProtocol Protocol

Summary

The AMQP protocol to be used

public ushort RequestedChannelMax

Summary

Maximum channel number to ask for

public int RequestedConnectionTimeout

Summary

Timeout setting for connection attempts (in milliseconds)

public uint RequestedFrameMax

Summary

Frame-max parameter to ask for (in bytes)

public ushort RequestedHeartbeat

Summary

Heartbeat setting to request (in seconds)

public ConnectionFactory.ObtainSocket SocketFactory

Summary

Set custom socket options by providing a SocketFactory

public SslOption Ssl

Summary

Ssl options setting

public string UserName

Summary

Username to use when authenticating to the server

public string VirtualHost

Summary

Virtual host to access during this connection

Property Detail

public AmqpTcpEndpoint Endpoint (rw)

Summary

The AMQP connection target

public Uri uri (w)

Summary

Set connection parameters using the amqp or amqps scheme

public string Uri (w)

Summary

Set connection parameters using the amqp or amqps scheme

Constructor Detail

ConnectionFactory

```
public ConnectionFactory()
```

Summary

Construct a fresh instance, with all fields set to their respective defaults.

Method Detail

AuthMechanismFactory

```
public AuthMechanismFactory AuthMechanismFactory(string[] mechs)
```

Flags public
Return type AuthMechanismFactory

Parameters	Name	Type
	mechs	string[]

Summary

Given a list of mechanism names supported by the server, select a preferred mechanism, or null if we have none in common.

CreateConnection

```
public virtual IConnection CreateConnection()
```

Flags	public virtual
Return type	IConnection

Summary

Create a connection to the specified endpoint No broker-originated redirects are permitted.

CreateConnection

```
public virtual IConnection CreateConnection(int maxRedirects)
```

Flags	public virtual
Return type	IConnection

Parameters	Name	Type
	maxRedirects	int

Summary

Create a connection to the first available endpoint in the list provided. Up to a maximum of maxRedirects broker-originated redirects are permitted for each endpoint tried.

DefaultSocketFactory

```
public static TcpClient DefaultSocketFactory(AddressFamily addressFamily)
```

Flags	public static
Return type	TcpClient

Parameters	Name	Type
	addressFamily	AddressFamily

Index | Namespace RabbitMQ.Client

public class PlainMechanism

- implements AuthMechanism

Constructor Summary

Flags	Name	Summary
public	PlainMechanism()	(undocumented)

Method Summary

Flags	Name	Summary
public virtual final	byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)	(undocumented)

Constructor Detail

PlainMechanism

```
public PlainMechanism()
```

Method Detail

handleChallenge

```
public virtual final byte[] handleChallenge(byte[] challenge, ConnectionFactory factory)
```

Flags	public virtual final	
Return type	byte[]	
	Name	**Type**
Parameters	challenge	byte[]
	factory	ConnectionFactory

Index | Namespace RabbitMQ.Client

public class PlainMechanismFactory

- implements AuthMechanismFactory

Property Summary

Flags	Type	Name	Summary
public virtual final	string	Name (r)	(undocumented)

Constructor Summary

Flags	Name	Summary
public	PlainMechanismFactory()	(undocumented)

Method Summary

Flags	Name	Summary
public virtual final	AuthMechanism GetInstance()	(undocumented)

Property Detail

public virtual final string Name (r)

Constructor Detail

PlainMechanismFactory

```
public PlainMechanismFactory()
```

Method Detail

GetInstance

```
public virtual final AuthMechanism GetInstance()
```

Flags	public virtual final
Return type	AuthMechanism

public class Protocols

Summary

Concrete, predefined IProtocol instances ready for use with ConnectionFactory.

Remarks

Applications will in the common case use the FromEnvironment() method to search a fallback-chain of configuration sources for the IProtocol instance to use. However, in some cases, the default fallback-chain is not appropriate; in these cases, other methods such as FromConfiguration(string) or SafeLookup(string) may suffice.

Field Summary

Flags	Type	Name	Summary
public initonly static	`string`	DefaultAppSettingsKey	The default App.config appSettings key used by FromConfiguration and FromEnvironment. At the time of writing, "AMQP_PROTOCOL".
public initonly static	`string`	EnvironmentVariable	The environment variable read by FromEnvironmentVariable() and FromEnvironment(). At the time of writing, "AMQP_PROTOCOL".

Property Summary

Flags	Type	Name	Summary
public static	IProtocol	AMQP_0_8 (r)	Protocol version 0-8 as standardised.
public static	IProtocol	AMQP_0_8_QPID (r)	Protocol version 0-8, as modified by QPid.
public static	IProtocol	AMQP_0_9 (r)	Protocol version 0-9 as standardised (omitting sections marked "WIP", "work in progress", including in particular the Message class of operations).
public static	IProtocol	AMQP_0_9_1 (r)	Protocol version 0-9-1 as modified by VMWare.
public static	IProtocol	DefaultProtocol (r)	Retrieve the current default protocol variant (currently AMQP_0_9_1)

Method Summary

Flags	Name	Summary
public static	IProtocol FromConfiguration(string appSettingsKey)	Uses App.config's appSettings section to retrieve an IProtocol instance.
public static	IProtocol FromConfiguration()	Returns FromConfiguration(DefaultAppSettingsKey).
public static	IProtocol FromEnvironment()	Returns FromEnvironment(DefaultAppSettingsKey).
public static	IProtocol FromEnvironment(string appSettingsKey)	Tries FromConfiguration() first, followed by FromEnvironmentVariable() if no setting was found in the App.config.
public static	IProtocol FromEnvironmentVariable()	Uses the process environment variable `EnvironmentVariable` to retrieve an IProtocol instance.
public static	IProtocol Lookup(string name)	Low-level method for retrieving a protocol version by name (of one of the static properties on this class)
public static	IProtocol SafeLookup(string name)	Retrieve a protocol version by name (of one of the static properties on this class)

Field Detail

public initonly static string DefaultAppSettingsKey

Summary

The default App.config appSettings key used by FromConfiguration and FromEnvironment. At the time of writing, "AMQP_PROTOCOL".

public initonly static string EnvironmentVariable

Summary

The environment variable read by FromEnvironmentVariable() and FromEnvironment(). At the time of writing, "AMQP_PROTOCOL".

Property Detail

public static IProtocol AMQP_0_8 (r)

Summary

Protocol version 0-8 as standardised.

public static IProtocol AMQP_0_8_QPID (r)

Summary

Protocol version 0-8, as modified by QPid.

public static IProtocol AMQP_0_9 (r)

Summary

Protocol version 0-9 as standardised (omitting sections marked "WIP", "work in progress", including in particular the Message class of operations).

public static IProtocol AMQP_0_9_1 (r)

Summary

Protocol version 0-9-1 as modified by VMWare.

public static IProtocol DefaultProtocol (r)

Summary

Retrieve the current default protocol variant (currently AMQP_0_9_1)

Method Detail

FromConfiguration

```
public static IProtocol FromConfiguration(string appSettingsKey)
```

Flags public static
Return type IProtocol

Parameters	Name	Type
	appSettingsKey	`string`

Summary

Uses App.config's appSettings section to retrieve an IProtocol instance.

Remarks

If the appSettings key is missing, Protocols.DefaultProtocol is used. If the protocol variant named is not found, ConfigurationException is thrown.

Exception

FromConfiguration

```
public static IProtocol FromConfiguration()
```

Flags	public static
Return type	IProtocol

Summary

Returns FromConfiguration(DefaultAppSettingsKey).

FromEnvironment

```
public static IProtocol FromEnvironment()
```

Flags	public static
Return type	IProtocol

Summary

Returns FromEnvironment(DefaultAppSettingsKey).

FromEnvironment

```
public static IProtocol FromEnvironment(string appSettingsKey)
```

Flags	public static
Return type	IProtocol

Parameters	Name	Type
	appSettingsKey	`string`

Summary

Tries FromConfiguration() first, followed by FromEnvironmentVariable() if no setting was found in the App.config.

Exception

FromEnvironmentVariable

```
public static IProtocol FromEnvironmentVariable()
```

Flags	public static
Return type	IProtocol

Summary

Uses the process environment variable

```
EnvironmentVariable
```

to retrieve an IProtocol instance.

Remarks

If the environment variable is unset, Protocols.DefaultProtocol is used. If the protocol variant named is not found, ConfigurationException is thrown.

Exception

Lookup

```
public static IProtocol Lookup(string name)
```

Flags	public static
Return type	IProtocol
Parameters	**Name** **Type** name string

Summary

Low-level method for retrieving a protocol version by name (of one of the static properties on this class)

Remarks

Returns null if no suitable property could be found.

In many cases, FromEnvironment() will be a more appropriate method for applications to call; this method is provided for cases where the caller wishes to know the answer to the question "does a suitable IProtocol property with this name exist, and if so, what is its value?"

SafeLookup

```
public static IProtocol SafeLookup(string name)
```

Flags	public static
Return type	IProtocol
Parameters	**Name** **Type** name string

Summary

Retrieve a protocol version by name (of one of the static properties on this class)

Remarks

If the argument is null, Protocols.DefaultProtocol is used. If the protocol variant named is not found, ConfigurationException is thrown.

In many cases, FromEnvironment() will be a more appropriate method for applications to call; this method is provided for cases where the caller wishes to know the answer to the question "does a suitable IProtocol property with this name exist, and if so, what is its value?", with the additional guarantee that if a suitable property does not exist, a ConfigurationException will be thrown.

Exception

Index | Namespace RabbitMQ.Client

public class PublicationAddress

Summary

Container for an exchange name, exchange type and routing key, usable as the target address of a message to be published.

Remarks

The syntax used for the external representation of instances of this class is compatible with QPid's "Reply-To" field pseudo-URI format. The pseudo-URI format is (exchange-type):///(exchange-name)/(routing-key), where exchange-type is one of the permitted exchange type names (see class ExchangeType), exchange-name must be present but may be empty, and routing-key must be present but may be empty.

The syntax is as it is solely for compatibility with QPid's existing usage of the ReplyTo field; the AMQP specifications 0-8 and 0-9 do not define the format of the field, and do not define any format for the triple (exchange name, exchange type, routing key) that could be used instead. Please see also the way class RabbitMQ.Client.MessagePatterns.SimpleRpcServer uses the ReplyTo field.

Field Summary

Flags	Type	Name	Summary
public initonly static	`Regex`	`PSEUDO_URI_PARSER`	Regular expression used to extract the exchange-type, exchange-name and routing-key from a string.

Property Summary

Flags	Type	Name	Summary
public	`string`	`ExchangeName` (r)	Retrieve the exchange name.
public	`string`	`ExchangeType` (r)	Retrieve the exchange type string.
public	`string`	`RoutingKey` (r)	Retrieve the routing key.

Constructor Summary

Flags	Name	Summary
public	`PublicationAddress(string exchangeType, string exchangeName, string routingKey)`	Construct a PublicationAddress with the given exchange type, exchange name and routing key.

Method Summary

Flags	Name	Summary
public static	`PublicationAddress Parse(string uriLikeString)`	Parse a PublicationAddress out of the given string, using the PSEUDO_URI_PARSER regex.
public virtual	`string ToString()`	Reconstruct the "uri" from its constituents.

Field Detail

public initonly static Regex PSEUDO_URI_PARSER

Summary

Regular expression used to extract the exchange-type, exchange-name and routing-key from a string.

Property Detail

public string ExchangeName (r)

Summary

Retrieve the exchange name.

public string ExchangeType (r)

Summary

Retrieve the exchange type string.

public string RoutingKey (r)

Summary

Retrieve the routing key.

Constructor Detail

PublicationAddress

```
public PublicationAddress(string exchangeType, string exchangeName, string routingKey)
```

	Name	Type
Parameters	exchangeType	`string`
	exchangeName	`string`
	routingKey	`string`

Summary

Construct a PublicationAddress with the given exchange type, exchange name and routing key.

Method Detail

Parse

```
public static PublicationAddress Parse(string uriLikeString)
```

Flags	public static
Return type	PublicationAddress

	Name	Type
Parameters	uriLikeString	`string`

Summary

Parse a PublicationAddress out of the given string, using the PSEUDO_URI_PARSER regex.

ToString

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

Flags	public virtual
Return type	`string`

Summary

Reconstruct the "uri" from its constituents.

Index | Namespace RabbitMQ.Client

public class QueueDeclareOk

Property Summary

Flags	Type	Name	Summary
public	`uint`	ConsumerCount (rw)	(undocumented)
public	`uint`	MessageCount (rw)	(undocumented)
public	`string`	QueueName (rw)	(undocumented)

Constructor Summary

Flags	Name	Summary
public	QueueDeclareOk(string queueName, uint messageCount, uint consumerCount)	(undocumented)

Property Detail

public uint ConsumerCount (rw)

public uint MessageCount (rw)

public string QueueName (rw)

Constructor Detail

QueueDeclareOk

```
public QueueDeclareOk(string queueName, uint messageCount, uint consumerCount)
```

	Name	Type
Parameters	queueName	`string`
	messageCount	`uint`
	consumerCount	`uint`

Index | Namespace RabbitMQ.Client

public class QueueingBasicConsumer

- extends DefaultBasicConsumer

Summary

Simple IBasicConsumer implementation that uses a SharedQueue to buffer incoming deliveries.

Remarks

Received messages are placed in the SharedQueue as instances of BasicDeliverEventArgs.

Note that messages taken from the SharedQueue may need acknowledging with IModel.BasicAck.

When the consumer is closed, through BasicCancel or through the shutdown of the underlying IModel or IConnection, the SharedQueue's Close() method is called, which causes any Enqueue() operations, and Dequeue() operations when the queue is empty, to throw EndOfStreamException (see the comment for SharedQueue.Close()).

The following is a simple example of the usage of this class:

```
IModel channel = ...;
QueueingBasicConsumer consumer = new QueueingBasicConsumer(channel);
channel.BasicConsume(queueName, null, consumer);

// At this point, messages will be being asynchronously delivered,
// and will be queueing up in consumer.Queue.

while (true) {
    try {
        BasicDeliverEventArgs e = (BasicDeliverEventArgs) consumer.Queue.Dequeue();
        // ... handle the delivery ...
        channel.BasicAck(e.DeliveryTag, false);
    } catch (EndOfStreamException ex) {
        // The consumer was cancelled, the model closed, or the
        // connection went away.
        break;
    }
}
```

Property Summary

Flags	Type	Name	Summary
public	SharedQueue	Queue (r)	Retrieves the SharedQueue that messages arrive on.

Constructor Summary

Flags	Name	Summary
public	QueueingBasicConsumer(IModel model, SharedQueue queue)	Creates a fresh QueueingBasicConsumer, initialising the Model and Queue properties to the given values.
public	QueueingBasicConsumer(IModel model)	Creates a fresh QueueingBasicConsumer, with Model set to the argument, and Queue set to a fresh SharedQueue.
public	QueueingBasicConsumer()	Creates a fresh QueueingBasicConsumer, initialising the Model property to null and the Queue property to a fresh SharedQueue.

Method Summary

Flags	Name	Summary
public virtual	void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool redelivered, string exchange, string routingKey, IBasicProperties properties, byte[] body)	Overrides DefaultBasicConsumer's HandleBasicDeliver implementation, building a

		BasicDeliverEventArgs instance and placing it in the Queue.
public virtual	void OnCancel()	Overrides DefaultBasicConsumer's OnCancel implementation, extending it to call the Close() method of the SharedQueue.

Property Detail

public SharedQueue Queue (r)

Summary

Retrieves the SharedQueue that messages arrive on.

Constructor Detail

QueueingBasicConsumer

```
public QueueingBasicConsumer(IModel model, SharedQueue queue)
```

	Name	Type
Parameters	model	IModel
	queue	SharedQueue

Summary

Creates a fresh QueueingBasicConsumer, initialising the Model and Queue properties to the given values.

QueueingBasicConsumer

```
public QueueingBasicConsumer(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Creates a fresh QueueingBasicConsumer, with Model set to the argument, and Queue set to a fresh SharedQueue.

QueueingBasicConsumer

```
public QueueingBasicConsumer()
```

Summary

Creates a fresh QueueingBasicConsumer, initialising the Model property to null and the Queue property to a fresh SharedQueue.

Method Detail

HandleBasicDeliver

```
public virtual void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool
redelivered, string exchange, string routingKey, IBasicProperties properties, byte[]
body)
```

Flags	public virtual
Return type	`void`

Parameters

Name	Type
consumerTag	`string`
deliveryTag	`ulong`
redelivered	`bool`
exchange	`string`
routingKey	`string`
properties	`IBasicProperties`
body	`byte[]`

Summary

Overrides DefaultBasicConsumer's HandleBasicDeliver implementation, building a BasicDeliverEventArgs instance and placing it in the Queue.

OnCancel

```
public virtual void OnCancel()
```

Flags	public virtual
Return type	`void`

Summary

Overrides DefaultBasicConsumer's OnCancel implementation, extending it to call the Close() method of the SharedQueue.

Index | Namespace RabbitMQ.Client

public class ShutdownEventArgs

- extends `EventArgs`

Summary

Information about the reason why a particular model, session, or connection was destroyed.

Remarks

The ClassId and Initiator properties should be used to determine the originator of the shutdown event.

Property Summary

Flags	Type	Name	Summary
public	`object`	`Cause` (r)	Object causing the shutdown, or null if none.
public	`ushort`	`ClassId` (r)	AMQP content-class ID, or 0 if none.
public	`ShutdownInitiator`	`Initiator` (r)	Returns the source of the shutdown event: either the application, the library, or the remote peer.
public	`ushort`	`MethodId` (r)	AMQP method ID within a content-class, or 0 if none.
public	`ushort`	`ReplyCode` (r)	One of the standardised AMQP reason codes. See RabbitMQ.Client.Framing.*.Constants.
public	`string`	`ReplyText` (r)	Informative human-readable reason text.

Constructor Summary

Flags	Name	Summary
public	`ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText, ushort classId, ushort methodId)`	Construct a ShutdownEventArgs with the given parameters and a null cause.
public	`ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText, ushort classId, ushort methodId, object cause)`	Construct a ShutdownEventArgs with the given parameters.
public	`ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText)`	Construct a ShutdownEventArgs with the given parameters, 0 for ClassId and MethodId, and a null Cause.
public	`ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText, object cause)`	Construct a ShutdownEventArgs with the given parameters and 0 for ClassId and MethodId.

Method Summary

Flags	Name	Summary
public virtual	`string ToString()`	Override ToString to be useful for debugging.

Property Detail

public object Cause (r)

Summary

Object causing the shutdown, or null if none.

public ushort ClassId (r)

Summary

AMQP content-class ID, or 0 if none.

public ShutdownInitiator Initiator (r)

Summary

Returns the source of the shutdown event: either the application, the library, or the remote peer.

public ushort MethodId (r)

Summary

AMQP method ID within a content-class, or 0 if none.

public ushort ReplyCode (r)

Summary

One of the standardised AMQP reason codes. See RabbitMQ.Client.Framing.*.Constants.

public string ReplyText (r)

Summary

Informative human-readable reason text.

Constructor Detail

ShutdownEventArgs

```
public ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText,
ushort classId, ushort methodId)
```

Parameters

Name	Type
initiator	ShutdownInitiator
replyCode	ushort
replyText	string
classId	ushort
methodId	ushort

Summary

Construct a ShutdownEventArgs with the given parameters and a null cause.

ShutdownEventArgs

```
public ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText,
ushort classId, ushort methodId, object cause)
```

Parameters

Name	Type
initiator	ShutdownInitiator
replyCode	ushort
replyText	string
classId	ushort
methodId	ushort

	cause	`object`

Summary

Construct a ShutdownEventArgs with the given parameters.

ShutdownEventArgs

```
public ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText)
```

	Name	Type
Parameters	initiator	ShutdownInitiator
	replyCode	`ushort`
	replyText	`string`

Summary

Construct a ShutdownEventArgs with the given parameters, 0 for ClassId and MethodId, and a null Cause.

ShutdownEventArgs

```
public ShutdownEventArgs(ShutdownInitiator initiator, ushort replyCode, string replyText,
object cause)
```

	Name	Type
	initiator	ShutdownInitiator
Parameters	replyCode	`ushort`
	replyText	`string`
	cause	`object`

Summary

Construct a ShutdownEventArgs with the given parameters and 0 for ClassId and MethodId.

Method Detail

ToString

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

Flags	public virtual
Return type	`string`

Summary

Override ToString to be useful for debugging.

Index | Namespace RabbitMQ.Client

public enum struct ShutdownInitiator

- extends Enum

Summary

Describes the source of a shutdown event.

Field Summary

Flags	Type	Name	Summary
public const	ShutdownInitiator	Application	The shutdown event originated from the application using the RabbitMQ .NET client library.
public const	ShutdownInitiator	Library	The shutdown event originated from the RabbitMQ .NET client library itself.
public const	ShutdownInitiator	Peer	The shutdown event originated from the remote AMQP peer.

Field Detail

public const ShutdownInitiator Application

Summary

The shutdown event originated from the application using the RabbitMQ .NET client library.

public const ShutdownInitiator Library

Summary

The shutdown event originated from the RabbitMQ .NET client library itself.

Remarks

Shutdowns with this ShutdownInitiator code may appear if, for example, an internal error is detected by the client, or if the server sends a syntactically invalid frame. Another potential use is on IConnection AutoClose.

public const ShutdownInitiator Peer

Summary

The shutdown event originated from the remote AMQP peer.

Remarks

A valid received connection.close or channel.close event will manifest as a shutdown with this ShutdownInitiator.

Index | Namespace RabbitMQ.Client

public class ShutdownReportEntry

Summary

Single entry object in the shutdown report that encapsulates description of the error which occured during shutdown

Field Summary

Flags	Type	Name	Summary
public	`string`	`m_description`	(undocumented)
public	`Exception`	`m_ex`	(undocumented)

Property Summary

Flags	Type	Name	Summary
public	`string`	`Description` (r)	Description provided in the error
public	`Exception`	`Exception` (r)	Exception object that occured during shutdown, or null if unspecified

Constructor Summary

Flags	Name	Summary
public	`ShutdownReportEntry(string description, Exception ex)`	(undocumented)

Method Summary

Flags	Name	Summary
public virtual	`string ToString()`	(undocumented)

Field Detail

public string m_description

public Exception m_ex

Property Detail

public string Description (r)

Summary

Description provided in the error

public Exception Exception (r)

Summary

Exception object that occured during shutdown, or null if unspecified

Constructor Detail

ShutdownReportEntry

```
public ShutdownReportEntry(string description, Exception ex)
```

	Name	Type
Parameters	description	`string`
	ex	`Exception`

Method Detail

ToString

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

Flags	public virtual
Return type	`string`

Index | Namespace RabbitMQ.Client

public class SslHelper

Summary

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

Method Summary

Flags	Name	Summary
public static	`Stream TcpUpgrade(Stream tcpStream, SslOption sslOption)`	Upgrade a Tcp stream to an Ssl stream using the SSL options provided

Method Detail

TcpUpgrade

```
public static Stream TcpUpgrade(Stream tcpStream, SslOption sslOption)
```

Flags	public static	
Return type	`Stream`	
	Name	**Type**
Parameters	tcpStream	`Stream`
	sslOption	`SslOption`

Summary

Upgrade a Tcp stream to an Ssl stream using the SSL options provided

Index | Namespace RabbitMQ.Client

public class SslOption

Summary

Represents a configurable SSL option, used in setting up an SSL connection.

Property Summary

Flags	Type	Name	Summary
public	`SslPolicyErrors`	`AcceptablePolicyErrors` (rw)	Retrieve or set the set of ssl policy errors that are deemed acceptable
public	`string`	`CertPassphrase` (rw)	Retrieve or set the path to client certificate.
public	`string`	`CertPath` (rw)	Retrieve or set the path to client certificate.
public	`X509CertificateCollection`	`Certs` (rw)	Retrieve or set the X509CertificateCollection containing the client certificate. If no collection is set, the client will attempt to load one from the specified CertPath.
public	`bool`	`Enabled` (rw)	Flag specifying if Ssl should indeed be used
public	`string`	`ServerName` (rw)	Retrieve or set server's Canonical Name. This MUST match the CN on the Certificate else the SSL connection will fail
public	`SslProtocols`	`Version` (rw)	Retrieve or set the Ssl protocol version

Constructor Summary

Flags	Name	Summary
public	`SslOption()`	Construct an SslOption with no parameters set
public	`SslOption(string serverName)`	Construct an SslOption with just the server cannonical name. The Certificate path is set to an empty string
public	`SslOption(string serverName, string certPath, bool enabled)`	Construct an SslOption specifying both the server cannonical name and the client's certificate path.

Property Detail

public SslPolicyErrors AcceptablePolicyErrors (rw)

Summary

Retrieve or set the set of ssl policy errors that are deemed acceptable

public string CertPassphrase (rw)

Summary

Retrieve or set the path to client certificate.

public string CertPath (rw)

Summary

Retrieve or set the path to client certificate.

public X509CertificateCollection Certs (rw)

Summary

Retrieve or set the X509CertificateCollection containing the client certificate. If no collection is set, the client will attempt to load one from the specified CertPath.

public bool Enabled (rw)

Summary

Flag specifying if Ssl should indeed be used

public string ServerName (rw)

Summary

Retrieve or set server's Canonical Name. This MUST match the CN on the Certificate else the SSL connection will fail

public SslProtocols Version (rw)

Summary

Retrieve or set the Ssl protocol version

Constructor Detail

SslOption

```
public SslOption()
```

Summary

Construct an SslOption with no parameters set

SslOption

```
public SslOption(string serverName)
```

	Name	Type
Parameters	serverName	`string`

Summary

Construct an SslOption with just the server cannonical name. The Certificate path is set to an empty string

SslOption

```
public SslOption(string serverName, string certPath, bool enabled)
```

	Name	Type
Parameters	serverName	`string`
	certPath	`string`
	enabled	`bool`

Summary

Construct an SslOption specifying both the server cannonical name and the client's certificate path.
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	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 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.Content

public class BasicMessageReader

- implements IMessageReader

Summary

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

Property Summary

Flags	Type	Name	Summary
public virtual final	byte[]	BodyBytes (r)	Implement IMessageReader.BodyBytes
public virtual final	Stream	BodyStream (r)	Implement IMessageReader.BodyStream
public virtual final	IDictionary	Headers (r)	Implement IMessageReader.Headers
public	IBasicProperties	Properties (r)	Retrieve the IBasicProperties associated with this instance.
public	NetworkBinaryReader	Reader (r)	Retrieve this instance's NetworkBinaryReader reading from BodyBytes.

Constructor Summary

Flags	Name	Summary
public	BasicMessageReader(IBasicProperties properties, byte[] body)	Construct an instance ready for reading.

Method Summary

Flags	Name	Summary
public virtual final	int RawRead(byte[] target, int offset, int length)	Implement IMessageReader.RawRead
public virtual final	int RawRead()	Implement IMessageReader.RawRead

Property Detail

public virtual final byte[] BodyBytes (r)

Summary

Implement IMessageReader.BodyBytes

public virtual final Stream BodyStream (r)

Summary

Implement IMessageReader.BodyStream

public virtual final IDictionary Headers (r)

Summary

Implement IMessageReader.Headers

public IBasicProperties Properties (r)

Summary

Retrieve the IBasicProperties associated with this instance.

public NetworkBinaryReader Reader (r)

Summary

Retrieve this instance's NetworkBinaryReader reading from BodyBytes.

Remarks

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

Constructor Detail

BasicMessageReader

```
public BasicMessageReader(IBasicProperties properties, byte[] body)
```

	Name	Type
Parameters	properties	IBasicProperties
	body	byte[]

Summary

Construct an instance ready for reading.

Method Detail

RawRead

```
public virtual final int RawRead(byte[] target, int offset, int length)
```

Flags	public virtual final	
Return type	int	
	Name	**Type**
Parameters	target	byte[]
	offset	int
	length	int

Summary

Implement IMessageReader.RawRead

RawRead

```
public virtual final int RawRead()
```

Flags	public virtual final
Return type	int

Summary

Implement IMessageReader.RawRead

Index | Namespace RabbitMQ.Client.Content

public class BytesMessageBuilder

- extends BasicMessageBuilder
- implements IBytesMessageBuilder

Summary

Constructs AMQP Basic-class messages binary-compatible with QPid's "BytesMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid BytesMessages.

Constructor Summary

Flags	Name	Summary
public	BytesMessageBuilder(IModel model, int initialAccumulatorSize)	Construct an instance for writing. See superclass.
public	BytesMessageBuilder(IModel model)	Construct an instance for writing. See superclass.

Method Summary

Flags	Name	Summary
public virtual	string GetDefaultContentType()	Override superclass method to answer our characteristic MIME type.
public virtual final	IBytesMessageBuilder Write(byte[] source, int offset, int count)	Write a section of a byte array into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteByte(byte value)	Writes a byte value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteBytes(byte[] source)	Write a byte array into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteChar(char value)	Writes a char value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteDouble(double value)	Writes a double value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteInt16(short value)	Writes a short value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteInt32(int value)	Writes an int value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteInt64(long value)	Writes a long value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteSingle(single value)	Writes a float value into the message body being assembled.
public virtual final	IBytesMessageBuilder WriteString(string value)	Writes a string value into the message body being assembled.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid BytesMessages.

Constructor Detail

BytesMessageBuilder

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

	Name	Type
Parameters	model	IModel
	initialAccumulatorSize	int

Summary

Construct an instance for writing. See superclass.

BytesMessageBuilder

```
public BytesMessageBuilder(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Construct an instance for writing. See superclass.

Method Detail

GetDefaultContentType

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

Flags	public virtual
Return type	string

Summary

Override superclass method to answer our characteristic MIME type.

Write

```
public virtual final IBytesMessageBuilder Write(byte[] source, int offset, int count)
```

Flags	public virtual final
Return type	IBytesMessageBuilder

	Name	Type
Parameters	source	byte[]
	offset	int
	count	int

Summary

Write a section of a byte array into the message body being assembled.

WriteByte

```
public virtual final IBytesMessageBuilder WriteByte(byte value)
```

Flags	public virtual final

Return type IBytesMessageBuilder

Parameters

Name	Type
value	byte

Summary

Writes a byte value into the message body being assembled.

WriteBytes

```
public virtual final IBytesMessageBuilder WriteBytes(byte[] source)
```

Flags public virtual final

Return type IBytesMessageBuilder

Parameters

Name	Type
source	byte[]

Summary

Write a byte array into the message body being assembled.

WriteChar

```
public virtual final IBytesMessageBuilder WriteChar(char value)
```

Flags public virtual final

Return type IBytesMessageBuilder

Parameters

Name	Type
value	char

Summary

Writes a char value into the message body being assembled.

WriteDouble

```
public virtual final IBytesMessageBuilder WriteDouble(double value)
```

Flags public virtual final

Return type IBytesMessageBuilder

Parameters

Name	Type
value	double

Summary

Writes a double value into the message body being assembled.

WriteInt16

```
public virtual final IBytesMessageBuilder WriteInt16(short value)
```

Flags public virtual final

Return type IBytesMessageBuilder

Parameters

Name	Type
value	short

Summary

Writes a short value into the message body being assembled.

WriteInt32

```
public virtual final IBytesMessageBuilder WriteInt32(int value)
```

Flags	public virtual final
Return type	IBytesMessageBuilder

Parameters

Name	Type
value	int

Summary

Writes an int value into the message body being assembled.

WriteInt64

```
public virtual final IBytesMessageBuilder WriteInt64(long value)
```

Flags	public virtual final
Return type	IBytesMessageBuilder

Parameters

Name	Type
value	long

Summary

Writes a long value into the message body being assembled.

WriteSingle

```
public virtual final IBytesMessageBuilder WriteSingle(single value)
```

Flags	public virtual final
Return type	IBytesMessageBuilder

Parameters

Name	Type
value	single

Summary

Writes a float value into the message body being assembled.

WriteString

```
public virtual final IBytesMessageBuilder WriteString(string value)
```

Flags	public virtual final
Return type	IBytesMessageBuilder

Parameters

Name	Type
value	string

Summary

Writes a string value into the message body being assembled.
Index | Namespace RabbitMQ.Client.Content

public class BytesMessageReader

- extends BasicMessageReader
- implements IBytesMessageReader

Summary

Analyzes AMQP Basic-class messages binary-compatible with QPid's "BytesMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid BytesMessages.

Constructor Summary

Flags	Name	Summary
public	BytesMessageReader(IBasicProperties properties, byte[] payload)	Construct an instance for reading. See superclass.

Method Summary

Flags	Name	Summary
public virtual final	int Read(byte[] target, int offset, int count)	Reads a given number ("count") of bytes from the message body, placing them into "target", starting at "offset".
public virtual final	byte ReadByte()	Reads a byte from the message body.
public virtual final	byte[] ReadBytes(int count)	Reads a given number of bytes from the message body.
public virtual final	char ReadChar()	Reads a char from the message body.
public virtual final	double ReadDouble()	Reads a double from the message body.
public virtual final	short ReadInt16()	Reads a short from the message body.
public virtual final	int ReadInt32()	Reads an int from the message body.
public virtual final	long ReadInt64()	Reads a long from the message body.
public virtual final	single ReadSingle()	Reads a float from the message body.
public virtual final	string ReadString()	Reads a string from the message body.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid BytesMessages.

Constructor Detail

BytesMessageReader

```
public BytesMessageReader(IBasicProperties properties, byte[] payload)
```

	Name	Type
Parameters	properties	IBasicProperties
	payload	byte[]

Summary

Construct an instance for reading. See superclass.

Method Detail

Read

```
public virtual final int Read(byte[] target, int offset, int count)
```

Flags	public virtual final
Return type	int

	Name	Type
Parameters	target	byte[]
	offset	int
	count	int

Summary

Reads a given number ("count") of bytes from the message body, placing them into "target", starting at "offset".

ReadByte

```
public virtual final byte ReadByte()
```

Flags	public virtual final
Return type	byte

Summary

Reads a byte from the message body.

ReadBytes

```
public virtual final byte[] ReadBytes(int count)
```

Flags	public virtual final
Return type	byte[]

	Name	Type
Parameters	count	int

Summary

Reads a given number of bytes from the message body.

ReadChar

```
public virtual final char ReadChar()
```

Flags	public virtual final

Return type `char`

Summary

Reads a char from the message body.

ReadDouble

```
public virtual final double ReadDouble()
```

Flags public virtual final

Return type `double`

Summary

Reads a double from the message body.

ReadInt16

```
public virtual final short ReadInt16()
```

Flags public virtual final

Return type `short`

Summary

Reads a short from the message body.

ReadInt32

```
public virtual final int ReadInt32()
```

Flags public virtual final

Return type `int`

Summary

Reads an int from the message body.

ReadInt64

```
public virtual final long ReadInt64()
```

Flags public virtual final

Return type `long`

Summary

Reads a long from the message body.

ReadSingle

```
public virtual final single ReadSingle()
```

Flags public virtual final

Return type `single`

Summary

Reads a float from the message body.

ReadString

```
public virtual final string ReadString()
```

Flags public virtual final

Return type `string`

Summary

Reads a string from the message body.
Index | Namespace RabbitMQ.Client.Content

public class BytesWireFormatting

Summary

Internal support class for use in reading and writing information binary-compatible with QPid's "BytesMessage" wire encoding.

Constructor Summary

Flags	Name	Summary
public	BytesWireFormatting()	(undocumented)

Method Summary

Flags	Name	Summary
public static	int Read(NetworkBinaryReader reader, byte[] target, int offset, int count)	(undocumented)
public static	byte ReadByte(NetworkBinaryReader reader)	(undocumented)
public static	byte[] ReadBytes(NetworkBinaryReader reader, int count)	(undocumented)
public static	char ReadChar(NetworkBinaryReader reader)	(undocumented)
public static	double ReadDouble(NetworkBinaryReader reader)	(undocumented)
public static	short ReadInt16(NetworkBinaryReader reader)	(undocumented)
public static	int ReadInt32(NetworkBinaryReader reader)	(undocumented)
public static	long ReadInt64(NetworkBinaryReader reader)	(undocumented)
public static	single ReadSingle(NetworkBinaryReader reader)	(undocumented)
public static	string ReadString(NetworkBinaryReader reader)	(undocumented)
public static	void Write(NetworkBinaryWriter writer, byte[] source, int offset, int count)	(undocumented)
public static	void WriteByte(NetworkBinaryWriter writer, byte value)	(undocumented)
public static	void WriteBytes(NetworkBinaryWriter writer, byte[] source)	(undocumented)
public static	void WriteChar(NetworkBinaryWriter writer, char value)	(undocumented)
public static	void WriteDouble(NetworkBinaryWriter writer, double value)	(undocumented)
public static	void WriteInt16(NetworkBinaryWriter writer, short value)	(undocumented)
public static	void WriteInt32(NetworkBinaryWriter writer, int value)	(undocumented)
public static	void WriteInt64(NetworkBinaryWriter writer, long value)	(undocumented)
public static	void WriteSingle(NetworkBinaryWriter writer, single value)	(undocumented)
public static	void WriteString(NetworkBinaryWriter writer, string value)	(undocumented)

Constructor Detail

BytesWireFormatting

```
public BytesWireFormatting()
```

Method Detail

Read

```
public static int Read(NetworkBinaryReader reader, byte[] target, int offset, int count)
```

Flags	public static	
Return type	`int`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader
	target	`byte[]`
	offset	`int`
	count	`int`

ReadByte

```
public static byte ReadByte(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`byte`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadBytes

```
public static byte[] ReadBytes(NetworkBinaryReader reader, int count)
```

Flags	public static	
Return type	`byte[]`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader
	count	`int`

ReadChar

```
public static char ReadChar(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`char`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadDouble

```
public static double ReadDouble(NetworkBinaryReader reader)
```

Flags	public static
Return type	double

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadInt16

```
public static short ReadInt16(NetworkBinaryReader reader)
```

Flags	public static
Return type	short

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadInt32

```
public static int ReadInt32(NetworkBinaryReader reader)
```

Flags	public static
Return type	int

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadInt64

```
public static long ReadInt64(NetworkBinaryReader reader)
```

Flags	public static
Return type	long

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadSingle

```
public static single ReadSingle(NetworkBinaryReader reader)
```

Flags	public static
Return type	single

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadString

```
public static string ReadString(NetworkBinaryReader reader)
```

Flags	public static
Return type	string

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

Write

```
public static void Write(NetworkBinaryWriter writer, byte[] source, int offset, int count)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	source	byte[]
	offset	int
	count	int

WriteByte

```
public static void WriteByte(NetworkBinaryWriter writer, byte value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	byte

WriteBytes

```
public static void WriteBytes(NetworkBinaryWriter writer, byte[] source)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	source	byte[]

WriteChar

```
public static void WriteChar(NetworkBinaryWriter writer, char value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	char

WriteDouble

```
public static void WriteDouble(NetworkBinaryWriter writer, double value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	double

WriteInt16

```
public static void WriteInt16(NetworkBinaryWriter writer, short value)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	short

WriteInt32

```
public static void WriteInt32(NetworkBinaryWriter writer, int value)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	int

WriteInt64

```
public static void WriteInt64(NetworkBinaryWriter writer, long value)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	long

WriteSingle

```
public static void WriteSingle(NetworkBinaryWriter writer, single value)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	single

WriteString

```
public static void WriteString(NetworkBinaryWriter writer, string value)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	string

Index | Namespace RabbitMQ.Client.Content

public interface IBytesMessageBuilder

- implements IMessageBuilder

Summary

Interface for constructing messages binary-compatible with QPid's "BytesMessage" wire encoding.

Method Summary

Name	Summary
IBytesMessageBuilder Write(byte[] source, int offset, int count)	Write a section of a byte array into the message body being assembled.
IBytesMessageBuilder WriteByte(byte value)	Writes a byte value into the message body being assembled.
IBytesMessageBuilder WriteBytes(byte[] source)	Write a byte array into the message body being assembled.
IBytesMessageBuilder WriteChar(char value)	Writes a char value into the message body being assembled.
IBytesMessageBuilder WriteDouble(double value)	Writes a double value into the message body being assembled.
IBytesMessageBuilder WriteInt16(short value)	Writes a short value into the message body being assembled.
IBytesMessageBuilder WriteInt32(int value)	Writes an int value into the message body being assembled.
IBytesMessageBuilder WriteInt64(long value)	Writes a long value into the message body being assembled.
IBytesMessageBuilder WriteSingle(single value)	Writes a float value into the message body being assembled.
IBytesMessageBuilder WriteString(string value)	Writes a string value into the message body being assembled.

Method Detail

Write

```
IBytesMessageBuilder Write(byte[] source, int offset, int count)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
source	byte[]
offset	int
count	int

Summary

Write a section of a byte array into the message body being assembled.

WriteByte

```
IBytesMessageBuilder WriteByte(byte value)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
value	byte

Summary

Writes a byte value into the message body being assembled.

WriteBytes

```
IBytesMessageBuilder WriteBytes(byte[] source)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
source	byte[]

Summary

Write a byte array into the message body being assembled.

WriteChar

```
IBytesMessageBuilder WriteChar(char value)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
value	char

Summary

Writes a char value into the message body being assembled.

WriteDouble

```
IBytesMessageBuilder WriteDouble(double value)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
value	double

Summary

Writes a double value into the message body being assembled.

WriteInt16

```
IBytesMessageBuilder WriteInt16(short value)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
value	short

Summary

Writes a short value into the message body being assembled.

WriteInt32

```
IBytesMessageBuilder WriteInt32(int value)
```

Return type IBytesMessageBuilder

Parameters

Name	Type
value	int

Summary

Writes an int value into the message body being assembled.

WriteInt64

```
IBytesMessageBuilder WriteInt64(long value)
```

Return type	IBytesMessageBuilder	
Parameters	**Name**	**Type**
	value	long

Summary

Writes a long value into the message body being assembled.

WriteSingle

```
IBytesMessageBuilder WriteSingle(single value)
```

Return type	IBytesMessageBuilder	
Parameters	**Name**	**Type**
	value	single

Summary

Writes a float value into the message body being assembled.

WriteString

```
IBytesMessageBuilder WriteString(string value)
```

Return type	IBytesMessageBuilder	
Parameters	**Name**	**Type**
	value	string

Summary

Writes a string value into the message body being assembled.
Index | Namespace RabbitMQ.Client.Content

public interface IBytesMessageReader

- implements IMessageReader

Summary

Analyzes messages binary-compatible with QPid's "BytesMessage" wire encoding.

Method Summary

Name	Summary
int Read(byte[] target, int offset, int count)	Reads a given number ("count") of bytes from the message body, placing them into "target", starting at "offset".
byte ReadByte()	Reads a byte from the message body.
byte[] ReadBytes(int count)	Reads a given number of bytes from the message body.
char ReadChar()	Reads a char from the message body.
double ReadDouble()	Reads a double from the message body.
short ReadInt16()	Reads a short from the message body.
int ReadInt32()	Reads an int from the message body.
long ReadInt64()	Reads a long from the message body.
single ReadSingle()	Reads a float from the message body.
string ReadString()	Reads a string from the message body.

Method Detail

Read

```
int Read(byte[] target, int offset, int count)
```

Return type int

Parameters

Name	Type
target	byte[]
offset	int
count	int

Summary

Reads a given number ("count") of bytes from the message body, placing them into "target", starting at "offset".

ReadByte

```
byte ReadByte()
```

Return type byte

Summary

Reads a byte from the message body.

ReadBytes

```
byte[] ReadBytes(int count)
```

Return type byte[]

Parameters

Name	Type
count	int

Summary

Reads a given number of bytes from the message body.

ReadChar

```
char ReadChar()
```

Return type char
Summary

Reads a char from the message body.

ReadDouble

```
double ReadDouble()
```

Return type double
Summary

Reads a double from the message body.

ReadInt16

```
short ReadInt16()
```

Return type short
Summary

Reads a short from the message body.

ReadInt32

```
int ReadInt32()
```

Return type int
Summary

Reads an int from the message body.

ReadInt64

```
long ReadInt64()
```

Return type long
Summary

Reads a long from the message body.

ReadSingle

```
single ReadSingle()
```

Return type single
Summary

Reads a float from the message body.

ReadString

```
string ReadString()
```

Return type `string`

Summary

Reads a string from the message body.
Index | Namespace RabbitMQ.Client.Content

public interface IMapMessageBuilder

- implements IMessageBuilder

Summary

Interface for constructing messages binary-compatible with QPid's "MapMessage" wire encoding.

Property Summary

Type	Name	Summary
IDictionary	Body (r)	Retrieves the dictionary that will be written into the body of the message.

Property Detail

IDictionary Body (r)

Summary

Retrieves the dictionary that will be written into the body of the message.

Index | Namespace RabbitMQ.Client.Content

public interface IMapMessageReader

- implements IMessageReader

Summary

Analyzes messages binary-compatible with QPid's "MapMessage" wire encoding.

Property Summary

Type	Name	Summary
IDictionary	Body (r)	Parses the message body into an IDictionary instance.

Property Detail

IDictionary Body (r)

Summary

Parses the message body into an IDictionary instance.

Index | Namespace RabbitMQ.Client.Content

public interface IMessageBuilder

Summary

Interface for constructing application messages.

Remarks

Subinterfaces provide specialized data-writing methods. This base interface deals with the lowest common denominator: bytes, with no special encodings for higher-level objects.

Property Summary

Type	Name	Summary
`Stream`	`BodyStream` (r)	Retrieve the Stream being used to construct the message body.
`IDictionary`	`Headers` (r)	Retrieves the dictionary that will be used to construct the message header table.

Method Summary

Name	Summary
`byte[] GetContentBody()`	Finish and retrieve the content body for transmission.
`IContentHeader GetContentHeader()`	Finish and retrieve the content header for transmission.
`string GetDefaultContentType()`	Returns the default MIME content type for messages this instance constructs, or null if none is available or relevant.
`IMessageBuilder RawWrite(byte b)`	Write a single byte into the message body, without encoding or interpretation.
`IMessageBuilder RawWrite(byte[] bytes)`	Write a byte array into the message body, without encoding or interpretation.
`IMessageBuilder RawWrite(byte[] bytes, int offset, int length)`	Write a section of a byte array into the message body, without encoding or interpretation.

Property Detail

Stream BodyStream (r)

Summary

Retrieve the Stream being used to construct the message body.

IDictionary Headers (r)

Summary

Retrieves the dictionary that will be used to construct the message header table.

Method Detail

GetContentBody

```
byte[] GetContentBody()
```

Return type `byte[]`

Summary

Finish and retrieve the content body for transmission.

GetContentHeader

```
IContentHeader GetContentHeader()
```

Return type IContentHeader

Summary

Finish and retrieve the content header for transmission.

GetDefaultContentType

```
string GetDefaultContentType()
```

Return type string

Summary

Returns the default MIME content type for messages this instance constructs, or null if none is available or relevant.

RawWrite

```
IMessageBuilder RawWrite(byte b)
```

Return type IMessageBuilder

Parameters

Name	Type
b	byte

Summary

Write a single byte into the message body, without encoding or interpretation.

RawWrite

```
IMessageBuilder RawWrite(byte[] bytes)
```

Return type IMessageBuilder

Parameters

Name	Type
bytes	byte[]

Summary

Write a byte array into the message body, without encoding or interpretation.

RawWrite

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

Return type IMessageBuilder

Parameters

Name	Type
bytes	byte[]
offset	int
length	int

Summary

Write a section of a byte array into the message body, without encoding or interpretation.

Index | Namespace RabbitMQ.Client.Content

public interface IMessageReader

Summary

Interface for analyzing application messages.

Remarks

Subinterfaces provide specialized data-reading methods. This base interface deals with the lowest common denominator: bytes, with no special encodings for higher-level objects.

Property Summary

Type	Name	Summary
`byte[]`	`BodyBytes` (r)	Retrieve the message body, as a byte array.
`Stream`	`BodyStream` (r)	Retrieve the Stream being used to read from the message body.
`IDictionary`	`Headers` (r)	Retrieves the content header properties of the message being read.

Method Summary

Name	Summary
`int RawRead(byte[] target, int offset, int length)`	Read bytes from the body stream into a section of an existing byte array, without encoding or interpretation. Returns the number of bytes read from the body and written into the target array, which may be less than the number requested if the end-of-stream is reached.
`int RawRead()`	Read a single byte from the body stream, without encoding or interpretation. Returns -1 for end-of-stream.

Property Detail

byte[] BodyBytes (r)

Summary

Retrieve the message body, as a byte array.

Stream BodyStream (r)

Summary

Retrieve the Stream being used to read from the message body.

IDictionary Headers (r)

Summary

Retrieves the content header properties of the message being read.

Method Detail

RawRead

```
int RawRead(byte[] target, int offset, int length)
```

Return type `int`

Parameters

Name	Type
target	`byte[]`
offset	`int`

length `int`

Summary

Read bytes from the body stream into a section of an existing byte array, without encoding or interpretation. Returns the number of bytes read from the body and written into the target array, which may be less than the number requested if the end-of-stream is reached.

RawRead

```
int RawRead()
```

Return type `int`

Summary

Read a single byte from the body stream, without encoding or interpretation. Returns -1 for end-of-stream.

Index | Namespace RabbitMQ.Client.Content

public interface IStreamMessageBuilder

- implements IMessageBuilder

Summary

Interface for constructing messages binary-compatible with QPid's "StreamMessage" wire encoding.

Method Summary

Name	Summary
IStreamMessageBuilder WriteBool(bool value)	Writes a bool value into the message body being assembled.
IStreamMessageBuilder WriteByte(byte value)	Writes a byte value into the message body being assembled.
IStreamMessageBuilder WriteBytes(byte[] source, int offset, int count)	Writes a section of a byte array into the message body being assembled.
IStreamMessageBuilder WriteBytes(byte[] source)	Writes a byte array into the message body being assembled.
IStreamMessageBuilder WriteChar(char value)	Writes a char value into the message body being assembled.
IStreamMessageBuilder WriteDouble(double value)	Writes a double value into the message body being assembled.
IStreamMessageBuilder WriteInt16(short value)	Writes a short value into the message body being assembled.
IStreamMessageBuilder WriteInt32(int value)	Writes an int value into the message body being assembled.
IStreamMessageBuilder WriteInt64(long value)	Writes a long value into the message body being assembled.
IStreamMessageBuilder WriteObject(object value)	Writes an object value into the message body being assembled.
IStreamMessageBuilder WriteObjects(object[] values)	Writes objects using WriteObject(), one after the other. No length indicator is written. See also IStreamMessageReader.ReadObjects().
IStreamMessageBuilder WriteSingle(single value)	Writes a float value into the message body being assembled.
IStreamMessageBuilder WriteString(string value)	Writes a string value into the message body being assembled.

Method Detail

WriteBool

```
IStreamMessageBuilder WriteBool(bool value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	bool

Summary

Writes a bool value into the message body being assembled.

WriteByte

```
IStreamMessageBuilder WriteByte(byte value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	byte

Summary

Writes a byte value into the message body being assembled.

WriteBytes

```
IStreamMessageBuilder WriteBytes(byte[] source, int offset, int count)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
source	byte[]
offset	int
count	int

Summary

Writes a section of a byte array into the message body being assembled.

WriteBytes

```
IStreamMessageBuilder WriteBytes(byte[] source)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
source	byte[]

Summary

Writes a byte array into the message body being assembled.

WriteChar

```
IStreamMessageBuilder WriteChar(char value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	char

Summary

Writes a char value into the message body being assembled.

WriteDouble

```
IStreamMessageBuilder WriteDouble(double value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	double

Summary

Writes a double value into the message body being assembled.

WriteInt16

```
IStreamMessageBuilder WriteInt16(short value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	short

Summary

Writes a short value into the message body being assembled.

WriteInt32

```
IStreamMessageBuilder WriteInt32(int value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	int

Summary

Writes an int value into the message body being assembled.

WriteInt64

```
IStreamMessageBuilder WriteInt64(long value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	long

Summary

Writes a long value into the message body being assembled.

WriteObject

```
IStreamMessageBuilder WriteObject(object value)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
value	object

Summary

Writes an object value into the message body being assembled.

Remarks

The only permitted types are bool, int, short, byte, char, long, float, double, byte[] and string.

WriteObjects

```
IStreamMessageBuilder WriteObjects(object[] values)
```

Return type IStreamMessageBuilder

Parameters

Name	Type
values	object[]

Summary

Writes objects using WriteObject(), one after the other. No length indicator is written. See also IStreamMessageReader.ReadObjects().

WriteSingle

```
IStreamMessageBuilder WriteSingle(single value)
```

Return type	IStreamMessageBuilder	
Parameters	**Name**	**Type**
	value	single

Summary

Writes a float value into the message body being assembled.

WriteString

```
IStreamMessageBuilder WriteString(string value)
```

Return type	IStreamMessageBuilder	
Parameters	**Name**	**Type**
	value	string

Summary

Writes a string value into the message body being assembled.
Index | Namespace RabbitMQ.Client.Content

public interface IStreamMessageReader

- implements IMessageReader

Summary

Analyzes messages binary-compatible with QPid's "StreamMessage" wire encoding.

Method Summary

Name	Summary
bool ReadBool()	Reads a bool from the message body.
byte ReadByte()	Reads a byte from the message body.
byte[] ReadBytes()	Reads a byte array from the message body. The body contains information about the size of the array to read.
char ReadChar()	Reads a char from the message body.
double ReadDouble()	Reads a double from the message body.
short ReadInt16()	Reads a short from the message body.
int ReadInt32()	Reads an int from the message body.
long ReadInt64()	Reads a long from the message body.
object ReadObject()	Reads an object from the message body.
object[] ReadObjects()	Reads objects from the message body until the end-of-stream is reached.
single ReadSingle()	Reads a float from the message body.
string ReadString()	Reads a string from the message body.

Method Detail

ReadBool

```
bool ReadBool()
```

Return type bool

Summary

Reads a bool from the message body.

ReadByte

```
byte ReadByte()
```

Return type byte

Summary

Reads a byte from the message body.

ReadBytes

```
byte[] ReadBytes()
```

Return type byte[]

Summary

Reads a byte array from the message body. The body contains information about the size of the array to read.

ReadChar

```
char ReadChar()
```

Return type char
Summary

Reads a char from the message body.

ReadDouble

```
double ReadDouble()
```

Return type double
Summary

Reads a double from the message body.

ReadInt16

```
short ReadInt16()
```

Return type short
Summary

Reads a short from the message body.

ReadInt32

```
int ReadInt32()
```

Return type int
Summary

Reads an int from the message body.

ReadInt64

```
long ReadInt64()
```

Return type long
Summary

Reads a long from the message body.

ReadObject

```
object ReadObject()
```

Return type object
Summary

Reads an object from the message body.

ReadObjects

```
object[] ReadObjects()
```

Return type object[]

Summary

Reads objects from the message body until the end-of-stream is reached.

ReadSingle

```
single ReadSingle()
```

Return type `single`

Summary

Reads a float from the message body.

ReadString

```
string ReadString()
```

Return type `string`

Summary

Reads a string from the message body.
Index | Namespace RabbitMQ.Client.Content

public class MapMessageBuilder

- extends BasicMessageBuilder
- implements IMapMessageBuilder

Summary

Constructs AMQP Basic-class messages binary-compatible with QPid's "MapMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid MapMessages.

Property Summary

Flags	Type	Name	Summary
public virtual final	IDictionary	Body (r)	Implement IMapMessageBuilder.Body

Constructor Summary

Flags	Name	Summary
public	MapMessageBuilder(IModel model, int initialAccumulatorSize)	Construct an instance for writing. See superclass.
public	MapMessageBuilder(IModel model)	Construct an instance for writing. See superclass.

Method Summary

Flags	Name	Summary
public virtual	byte[] GetContentBody()	Override superclass method to write Body out into the message BodyStream before retrieving the final byte array.
public virtual	string GetDefaultContentType()	Override superclass method to answer our characteristic MIME type.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid MapMessages.

Property Detail

public virtual final IDictionary Body (r)

Summary

Implement IMapMessageBuilder.Body

Constructor Detail

MapMessageBuilder

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

	Name	Type
Parameters	model	IModel
	initialAccumulatorSize	int

Summary

Construct an instance for writing. See superclass.

MapMessageBuilder

```
public MapMessageBuilder(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Construct an instance for writing. See superclass.

Method Detail

GetContentBody

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

Flags public virtual

Return type byte[]

Summary

Override superclass method to write Body out into the message BodyStream before retrieving the final byte array.

Remarks

Calling this message clears Body to null. Subsequent calls will fault.

GetDefaultContentType

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

Flags public virtual

Return type string

Summary

Override superclass method to answer our characteristic MIME type.

Index | Namespace RabbitMQ.Client.Content

public class MapMessageReader

- extends BasicMessageReader
- implements IMapMessageReader

Summary

Analyzes AMQP Basic-class messages binary-compatible with QPid's "MapMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid MapMessages.

Property Summary

Flags	Type	Name	Summary
public virtual final	IDictionary	Body (r)	Implement IMapMessageReader.Body

Constructor Summary

Flags	Name	Summary
public	MapMessageReader(IBasicProperties properties, byte[] payload)	Construct an instance for reading. See superclass.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid MapMessages.

Property Detail

public virtual final IDictionary Body (r)

Summary

Implement IMapMessageReader.Body

Exception

Constructor Detail

MapMessageReader

```
public MapMessageReader(IBasicProperties properties, byte[] payload)
```

	Name	Type
Parameters	properties	IBasicProperties
	payload	byte[]

Summary

Construct an instance for reading. See superclass.

Index | Namespace RabbitMQ.Client.Content

public class MapWireFormatting

Summary

Internal support class for use in reading and writing information binary-compatible with QPid's "MapMessage" wire encoding.

Exception

Constructor Summary

Flags	Name	Summary
public	MapWireFormatting()	(undocumented)

Method Summary

Flags	Name	Summary
public static	IDictionary ReadMap(NetworkBinaryReader reader)	(undocumented)
public static	void WriteMap(NetworkBinaryWriter writer, IDictionary table)	(undocumented)

Constructor Detail

MapWireFormatting

```
public MapWireFormatting()
```

Method Detail

ReadMap

```
public static IDictionary ReadMap(NetworkBinaryReader reader)
```

Flags	public static	
Return type	IDictionary	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

WriteMap

```
public static void WriteMap(NetworkBinaryWriter writer, IDictionary table)
```

Flags	public static	
Return type	void	
Parameters	**Name**	**Type**
	writer	NetworkBinaryWriter
	table	IDictionary

Index | Namespace RabbitMQ.Client.Content

public class PrimitiveParser

Summary

Utility class for extracting typed values from strings.

Constructor Summary

Flags	Name	Summary
public	PrimitiveParser()	(undocumented)

Method Summary

Flags	Name	Summary
public static	void InvalidConversion(string target, object source)	Causes ProtocolViolationException to be thrown; called by the various "Parse*" methods when a syntax error is detected.
public static	bool ParseBool(string value)	Attempt to parse a string representation of a bool.
public static	byte ParseByte(string value)	Attempt to parse a string representation of a byte.
public static	double ParseDouble(string value)	Attempt to parse a string representation of a double.
public static	single ParseFloat(string value)	Attempt to parse a string representation of a float.
public static	int ParseInt(string value)	Attempt to parse a string representation of an int.
public static	long ParseLong(string value)	Attempt to parse a string representation of a long.
public static	short ParseShort(string value)	Attempt to parse a string representation of a short.

Constructor Detail

PrimitiveParser

```
public PrimitiveParser()
```

Method Detail

InvalidConversion

```
public static void InvalidConversion(string target, object source)
```

Flags	public static	
Return type	void	
	Name	**Type**
Parameters	target	string
	source	object

Summary

Causes ProtocolViolationException to be thrown; called by the various "Parse*" methods when a syntax error is detected.

Exception

ParseBool

```
public static bool ParseBool(string value)
```

Flags	public static
Return type	`bool`

Parameters	**Name**	**Type**
	value	`string`

Summary

Attempt to parse a string representation of a bool.

Exception

ParseByte

```
public static byte ParseByte(string value)
```

Flags	public static
Return type	`byte`

Parameters	**Name**	**Type**
	value	`string`

Summary

Attempt to parse a string representation of a byte.

Exception

ParseDouble

```
public static double ParseDouble(string value)
```

Flags	public static
Return type	`double`

Parameters	**Name**	**Type**
	value	`string`

Summary

Attempt to parse a string representation of a double.

Exception

ParseFloat

```
public static single ParseFloat(string value)
```

Flags	public static
Return type	`single`

Parameters	**Name**	**Type**
	value	`string`

Summary

Attempt to parse a string representation of a float.
Exception

ParseInt

```
public static int ParseInt(string value)
```

Flags	public static
Return type	`int`
Parameters	**Name** **Type** value `string`

Summary

Attempt to parse a string representation of an int.
Exception

ParseLong

```
public static long ParseLong(string value)
```

Flags	public static
Return type	`long`
Parameters	**Name** **Type** value `string`

Summary

Attempt to parse a string representation of a long.
Exception

ParseShort

```
public static short ParseShort(string value)
```

Flags	public static
Return type	`short`
Parameters	**Name** **Type** value `string`

Summary

Attempt to parse a string representation of a short.
Exception

Index | Namespace RabbitMQ.Client.Content

public class StreamMessageBuilder

- extends BasicMessageBuilder
- implements IStreamMessageBuilder

Summary

Constructs AMQP Basic-class messages binary-compatible with QPid's "StreamMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid StreamMessages.

Constructor Summary

Flags	Name	Summary
public	StreamMessageBuilder(IModel model, int initialAccumulatorSize)	Construct an instance for writing. See superclass.
public	StreamMessageBuilder(IModel model)	Construct an instance for writing. See superclass.

Method Summary

Flags	Name	Summary
public virtual	string GetDefaultContentType()	Override superclass method to answer our characteristic MIME type.
public virtual final	IStreamMessageBuilder WriteBool(bool value)	Writes a bool value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteByte(byte value)	Writes a byte value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteBytes(byte[] source)	Writes a byte array into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteBytes(byte[] source, int offset, int count)	Writes a section of a byte array into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteChar(char value)	Writes a char value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteDouble(double value)	Writes a double value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteInt16(short value)	Writes a short value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteInt32(int value)	Writes an int value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteInt64(long value)	Writes a long value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteObject(object value)	Writes an object value into the message body being assembled.

public virtual final	IStreamMessageBuilder WriteObjects(object[] values)	Writes objects using WriteObject(), one after the other. No length indicator is written. See also IStreamMessageReader.ReadObjects().
public virtual final	IStreamMessageBuilder WriteSingle(single value)	Writes a float value into the message body being assembled.
public virtual final	IStreamMessageBuilder WriteString(string value)	Writes a string value into the message body being assembled.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid StreamMessages.

Constructor Detail

StreamMessageBuilder

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

	Name	Type
Parameters	model	IModel
	initialAccumulatorSize	int

Summary

Construct an instance for writing. See superclass.

StreamMessageBuilder

```
public StreamMessageBuilder(IModel model)
```

	Name	Type
Parameters	model	IModel

Summary

Construct an instance for writing. See superclass.

Method Detail

GetDefaultContentType

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

Flags	public virtual
Return type	string

Summary

Override superclass method to answer our characteristic MIME type.

WriteBool

```
public virtual final IStreamMessageBuilder WriteBool(bool value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
value	bool

Summary

Writes a bool value into the message body being assembled.

WriteByte

```
public virtual final IStreamMessageBuilder WriteByte(byte value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
value	byte

Summary

Writes a byte value into the message body being assembled.

WriteBytes

```
public virtual final IStreamMessageBuilder WriteBytes(byte[] source)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
source	byte[]

Summary

Writes a byte array into the message body being assembled.

WriteBytes

```
public virtual final IStreamMessageBuilder WriteBytes(byte[] source, int offset, int
count)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
source	byte[]
offset	int
count	int

Summary

Writes a section of a byte array into the message body being assembled.

WriteChar

```
public virtual final IStreamMessageBuilder WriteChar(char value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters	Name	Type
	value	char

Summary

Writes a char value into the message body being assembled.

WriteDouble

```
public virtual final IStreamMessageBuilder WriteDouble(double value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters	Name	Type
	value	double

Summary

Writes a double value into the message body being assembled.

WriteInt16

```
public virtual final IStreamMessageBuilder WriteInt16(short value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters	Name	Type
	value	short

Summary

Writes a short value into the message body being assembled.

WriteInt32

```
public virtual final IStreamMessageBuilder WriteInt32(int value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters	Name	Type
	value	int

Summary

Writes an int value into the message body being assembled.

WriteInt64

```
public virtual final IStreamMessageBuilder WriteInt64(long value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters	Name	Type
	value	long

Summary

Writes a long value into the message body being assembled.

WriteObject

```
public virtual final IStreamMessageBuilder WriteObject(object value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
value	object

Summary

Writes an object value into the message body being assembled.

Remarks

The only permitted types are bool, int, short, byte, char, long, float, double, byte[] and string.

WriteObjects

```
public virtual final IStreamMessageBuilder WriteObjects(object[] values)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
values	object[]

Summary

Writes objects using WriteObject(), one after the other. No length indicator is written. See also IStreamMessageReader.ReadObjects().

WriteSingle

```
public virtual final IStreamMessageBuilder WriteSingle(single value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
value	single

Summary

Writes a float value into the message body being assembled.

WriteString

```
public virtual final IStreamMessageBuilder WriteString(string value)
```

Flags	public virtual final
Return type	IStreamMessageBuilder

Parameters

Name	Type
value	string

Summary

Writes a string value into the message body being assembled.

Index | Namespace RabbitMQ.Client.Content

public class StreamMessageReader

- extends BasicMessageReader
- implements IStreamMessageReader

Summary

Analyzes AMQP Basic-class messages binary-compatible with QPid's "StreamMessage" wire encoding.

Field Summary

Flags	Type	Name	Summary
public initonly static	string	MimeType	MIME type associated with QPid StreamMessages.

Constructor Summary

Flags	Name	Summary
public	StreamMessageReader(IBasicProperties properties, byte[] payload)	Construct an instance for reading. See superclass.

Method Summary

Flags	Name	Summary
public virtual final	bool ReadBool()	Reads a bool from the message body.
public virtual final	byte ReadByte()	Reads a byte from the message body.
public virtual final	byte[] ReadBytes()	Reads a byte array from the message body. The body contains information about the size of the array to read.
public virtual final	char ReadChar()	Reads a char from the message body.
public virtual final	double ReadDouble()	Reads a double from the message body.
public virtual final	short ReadInt16()	Reads a short from the message body.
public virtual final	int ReadInt32()	Reads an int from the message body.
public virtual final	long ReadInt64()	Reads a long from the message body.
public virtual final	object ReadObject()	Reads an object from the message body.
public virtual final	object[] ReadObjects()	Reads objects from the message body until the end-of-stream is reached.
public virtual final	single ReadSingle()	Reads a float from the message body.
public virtual final	string ReadString()	Reads a string from the message body.

Field Detail

public initonly static string MimeType

Summary

MIME type associated with QPid StreamMessages.

Constructor Detail

StreamMessageReader

```
public StreamMessageReader(IBasicProperties properties, byte[] payload)
```

	Name	Type
Parameters	properties	IBasicProperties
	payload	byte[]

Summary

Construct an instance for reading. See superclass.

Method Detail

ReadBool

```
public virtual final bool ReadBool()
```

Flags	public virtual final
Return type	bool

Summary

Reads a bool from the message body.

ReadByte

```
public virtual final byte ReadByte()
```

Flags	public virtual final
Return type	byte

Summary

Reads a byte from the message body.

ReadBytes

```
public virtual final byte[] ReadBytes()
```

Flags	public virtual final
Return type	byte[]

Summary

Reads a byte array from the message body. The body contains information about the size of the array to read.

ReadChar

```
public virtual final char ReadChar()
```

Flags	public virtual final
Return type	char

Summary

Reads a char from the message body.

ReadDouble

```
public virtual final double ReadDouble()
```

Flags public virtual final
Return type `double`
Summary

Reads a double from the message body.

ReadInt16

```
public virtual final short ReadInt16()
```

Flags public virtual final
Return type `short`
Summary

Reads a short from the message body.

ReadInt32

```
public virtual final int ReadInt32()
```

Flags public virtual final
Return type `int`
Summary

Reads an int from the message body.

ReadInt64

```
public virtual final long ReadInt64()
```

Flags public virtual final
Return type `long`
Summary

Reads a long from the message body.

ReadObject

```
public virtual final object ReadObject()
```

Flags public virtual final
Return type `object`
Summary

Reads an object from the message body.

ReadObjects

```
public virtual final object[] ReadObjects()
```

Flags public virtual final
Return type `object[]`
Summary

Reads objects from the message body until the end-of-stream is reached.

ReadSingle

```
public virtual final single ReadSingle()
```

Flags public virtual final
Return type `single`
Summary

Reads a float from the message body.

ReadString

```
public virtual final string ReadString()
```

Flags public virtual final
Return type `string`
Summary

Reads a string from the message body.
Index | Namespace RabbitMQ.Client.Content

public class StreamWireFormatting

Summary

Internal support class for use in reading and writing information binary-compatible with QPid's "StreamMessage" wire encoding.

Constructor Summary

Flags	Name	Summary
public	StreamWireFormatting()	(undocumented)

Method Summary

Flags	Name	Summary
public static	bool ReadBool(NetworkBinaryReader reader)	(undocumented)
public static	byte ReadByte(NetworkBinaryReader reader)	(undocumented)
public static	byte[] ReadBytes(NetworkBinaryReader reader)	(undocumented)
public static	char ReadChar(NetworkBinaryReader reader)	(undocumented)
public static	double ReadDouble(NetworkBinaryReader reader)	(undocumented)
public static	short ReadInt16(NetworkBinaryReader reader)	(undocumented)
public static	int ReadInt32(NetworkBinaryReader reader)	(undocumented)
public static	long ReadInt64(NetworkBinaryReader reader)	(undocumented)
public static	object ReadNonnullObject(string target, NetworkBinaryReader reader)	(undocumented)
public static	object ReadObject(NetworkBinaryReader reader)	(undocumented)
public static	single ReadSingle(NetworkBinaryReader reader)	(undocumented)
public static	string ReadString(NetworkBinaryReader reader)	(undocumented)
public static	string ReadUntypedString(NetworkBinaryReader reader)	(undocumented)
public static	void WriteBool(NetworkBinaryWriter writer, bool value)	(undocumented)
public static	void WriteByte(NetworkBinaryWriter writer, byte value)	(undocumented)
public static	void WriteBytes(NetworkBinaryWriter writer, byte[] value, int offset, int length)	(undocumented)
public static	void WriteBytes(NetworkBinaryWriter writer, byte[] value)	(undocumented)
public static	void WriteChar(NetworkBinaryWriter writer, char value)	(undocumented)
public static	void WriteDouble(NetworkBinaryWriter writer, double value)	(undocumented)
public static	void WriteInt16(NetworkBinaryWriter writer, short value)	(undocumented)
	void WriteInt32(NetworkBinaryWriter writer, int value)	(undocumented)

public static		
public static	void WriteInt64(NetworkBinaryWriter writer, long value)	(undocumented)
public static	void WriteObject(NetworkBinaryWriter writer, object value)	(undocumented)
public static	void WriteSingle(NetworkBinaryWriter writer, single value)	(undocumented)
public static	void WriteString(NetworkBinaryWriter writer, string value)	(undocumented)
public static	void WriteUntypedString(NetworkBinaryWriter writer, string value)	(undocumented)

Constructor Detail

StreamWireFormatting

```
public StreamWireFormatting()
```

Method Detail

ReadBool

```
public static bool ReadBool(NetworkBinaryReader reader)
```

Flags	public static
Return type	bool

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadByte

```
public static byte ReadByte(NetworkBinaryReader reader)
```

Flags	public static
Return type	byte

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadBytes

```
public static byte[] ReadBytes(NetworkBinaryReader reader)
```

Flags	public static
Return type	byte[]

Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadChar

```
public static char ReadChar(NetworkBinaryReader reader)
```

Flags	public static
Return type	char

Parameters	Name	Type
	reader	NetworkBinaryReader

ReadDouble

```
public static double ReadDouble(NetworkBinaryReader reader)
```

Flags	public static
Return type	double

Parameters	Name	Type
	reader	NetworkBinaryReader

ReadInt16

```
public static short ReadInt16(NetworkBinaryReader reader)
```

Flags	public static
Return type	short

Parameters	Name	Type
	reader	NetworkBinaryReader

ReadInt32

```
public static int ReadInt32(NetworkBinaryReader reader)
```

Flags	public static
Return type	int

Parameters	Name	Type
	reader	NetworkBinaryReader

ReadInt64

```
public static long ReadInt64(NetworkBinaryReader reader)
```

Flags	public static
Return type	long

Parameters	Name	Type
	reader	NetworkBinaryReader

ReadNonnullObject

```
public static object ReadNonnullObject(string target, NetworkBinaryReader reader)
```

Flags	public static
Return type	object

Parameters	Name	Type
	target	string
	reader	NetworkBinaryReader

Exception

ReadObject

```
public static object ReadObject(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`object`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

Exception

Exception

ReadSingle

```
public static single ReadSingle(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`single`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadString

```
public static string ReadString(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`string`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

ReadUntypedString

```
public static string ReadUntypedString(NetworkBinaryReader reader)
```

Flags	public static	
Return type	`string`	
Parameters	**Name**	**Type**
	reader	NetworkBinaryReader

WriteBool

```
public static void WriteBool(NetworkBinaryWriter writer, bool value)
```

Flags	public static	
Return type	`void`	
Parameters	**Name**	**Type**
	writer	NetworkBinaryWriter
	value	`bool`

WriteByte

```
public static void WriteByte(NetworkBinaryWriter writer, byte value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	byte

WriteBytes

```
public static void WriteBytes(NetworkBinaryWriter writer, byte[] value, int offset, int length)
```

Flags	public static
Return type	void

	Name	Type
	writer	NetworkBinaryWriter
Parameters	value	byte[]
	offset	int
	length	int

WriteBytes

```
public static void WriteBytes(NetworkBinaryWriter writer, byte[] value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	byte[]

WriteChar

```
public static void WriteChar(NetworkBinaryWriter writer, char value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	char

WriteDouble

```
public static void WriteDouble(NetworkBinaryWriter writer, double value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	double

WriteInt16

```
public static void WriteInt16(NetworkBinaryWriter writer, short value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	short

WriteInt32

```
public static void WriteInt32(NetworkBinaryWriter writer, int value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	int

WriteInt64

```
public static void WriteInt64(NetworkBinaryWriter writer, long value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	long

WriteObject

```
public static void WriteObject(NetworkBinaryWriter writer, object value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	object

Exception

WriteSingle

```
public static void WriteSingle(NetworkBinaryWriter writer, single value)
```

Flags	public static
Return type	void

	Name	Type
Parameters	writer	NetworkBinaryWriter
	value	single

WriteString

```
public static void WriteString(NetworkBinaryWriter writer, string value)
```

Flags	public static

Return type	`void`	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	`string`

WriteUntypedString

```
public static void WriteUntypedString(NetworkBinaryWriter writer, string value)
```

Flags	public static	
Return type	`void`	
	Name	**Type**
Parameters	writer	NetworkBinaryWriter
	value	`string`

Index | Namespace RabbitMQ.Client.Content

public enum struct StreamWireFormattingTag

- extends `Enum`

Summary

Tags used in parsing and generating StreamWireFormatting message bodies.

Field Summary

Flags	Type	Name	Summary
public const	StreamWireFormattingTag	Bool	(undocumented)
public const	StreamWireFormattingTag	Byte	(undocumented)
public const	StreamWireFormattingTag	Bytes	(undocumented)
public const	StreamWireFormattingTag	Char	(undocumented)
public const	StreamWireFormattingTag	Double	(undocumented)
public const	StreamWireFormattingTag	Int16	(undocumented)
public const	StreamWireFormattingTag	Int32	(undocumented)
public const	StreamWireFormattingTag	Int64	(undocumented)
public const	StreamWireFormattingTag	Null	(undocumented)
public const	StreamWireFormattingTag	Single	(undocumented)
public const	StreamWireFormattingTag	String	(undocumented)

Field Detail

public const StreamWireFormattingTag Bool

public const StreamWireFormattingTag Byte

public const StreamWireFormattingTag Bytes

public const StreamWireFormattingTag Char

public const StreamWireFormattingTag Double

public const StreamWireFormattingTag Int16

public const StreamWireFormattingTag Int32

public const StreamWireFormattingTag Int64

public const StreamWireFormattingTag Null

public const StreamWireFormattingTag Single

public const StreamWireFormattingTag String

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.
ConnectionShutdownEventHandler	Delegate used to process connection shutdown notifications.
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.Events

public delegate BasicAckEventHandler

- extends `MulticastDelegate`

```
public delegate void BasicAckEventHandler(IModel model, BasicAckEventArgs args)
```

Return type	`void`	
	Name	**Type**
Parameters	model	IModel
	args	BasicAckEventArgs

Summary

Delegate used to process Basic.Ack events.
Index | Namespace RabbitMQ.Client.Events

public class BasicDeliverEventArgs

- extends `EventArgs`

Summary

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

Property Summary

Flags	Type	Name	Summary
public	`IBasicProperties`	`BasicProperties` (rw)	The content header of the message.
public	`byte[]`	`Body` (rw)	The message body.
public	`string`	`ConsumerTag` (rw)	The consumer tag of the consumer that the message was delivered to.
public	`ulong`	`DeliveryTag` (rw)	The delivery tag for this delivery. See IModel.BasicAck.
public	`string`	`Exchange` (rw)	The exchange the message was originally published to.
public	`bool`	`Redelivered` (rw)	The AMQP "redelivered" flag.
public	`string`	`RoutingKey` (rw)	The routing key used when the message was originally published.

Constructor Summary

Flags	Name	Summary
public	`BasicDeliverEventArgs(string consumerTag, ulong deliveryTag, bool redelivered, string exchange, string routingKey, IBasicProperties properties, byte[] body)`	Constructor that fills the event's properties from its arguments.
public	`BasicDeliverEventArgs()`	Default constructor.

Property Detail

public IBasicProperties BasicProperties (rw)

Summary

The content header of the message.

public byte[] Body (rw)

Summary

The message body.

public string ConsumerTag (rw)

Summary

The consumer tag of the consumer that the message was delivered to.

public ulong DeliveryTag (rw)

Summary

The delivery tag for this delivery. See IModel.BasicAck.

public string Exchange (rw)

Summary

The exchange the message was originally published to.

public bool Redelivered (rw)

Summary

The AMQP "redelivered" flag.

public string RoutingKey (rw)

Summary

The routing key used when the message was originally published.

Constructor Detail

BasicDeliverEventArgs

```
public BasicDeliverEventArgs(string consumerTag, ulong deliveryTag, bool redelivered,
string exchange, string routingKey, IBasicProperties properties, byte[] body)
```

Parameters

Name	Type
consumerTag	string
deliveryTag	ulong
redelivered	bool
exchange	string
routingKey	string
properties	IBasicProperties
body	byte[]

Summary

Constructor that fills the event's properties from its arguments.

BasicDeliverEventArgs

```
public BasicDeliverEventArgs()
```

Summary

Default constructor.

Index | Namespace RabbitMQ.Client.Events

public delegate BasicDeliverEventHandler

- extends `MulticastDelegate`

```
public delegate void BasicDeliverEventHandler(IBasicConsumer sender,
BasicDeliverEventArgs args)
```

Return type	`void`	
	Name	**Type**
Parameters	sender	`IBasicConsumer`
	args	`BasicDeliverEventArgs`

Summary

Delegate used to process Basic.Deliver events.
Index | Namespace RabbitMQ.Client.Events

public class BasicNackEventArgs

- extends `EventArgs`

Summary

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

Property Summary

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

Constructor Summary

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

Property Detail

public ulong DeliveryTag (rw)

Summary

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

public bool Multiple (rw)

Summary

Whether this nack applies to one message or multiple messages.

public bool Requeue (rw)

Summary

Ignore

Remarks

Clients should ignore this field.

Constructor Detail

BasicNackEventArgs

`public BasicNackEventArgs()`

Summary

Default constructor.

Index | Namespace RabbitMQ.Client.Events

public delegate BasicNackEventHandler

- extends `MulticastDelegate`

```
public delegate void BasicNackEventHandler(IModel model, BasicNackEventArgs args)
```

Return type	`void`	
	Name	**Type**
Parameters	model	IModel
	args	BasicNackEventArgs

Summary

Delegate used to process Basic.Nack events.
Index | Namespace RabbitMQ.Client.Events

public delegate BasicRecoverOkEventHandler

- extends `MulticastDelegate`

```
public delegate void BasicRecoverOkEventHandler(IModel model, EventArgs args)
```

Return type `void`

	Name	Type
Parameters	model	IModel
	args	EventArgs

Summary

Delegate used to process Basic.RecoverOk events.
Index | Namespace RabbitMQ.Client.Events

public class BasicReturnEventArgs

- extends `EventArgs`

Summary

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

Property Summary

Flags	Type	Name	Summary
public	`IBasicProperties`	`BasicProperties` (rw)	The content header of the message.
public	`byte[]`	`Body` (rw)	The message body.
public	`string`	`Exchange` (rw)	The exchange the returned message was originally published to.
public	`ushort`	`ReplyCode` (rw)	The AMQP reason code for the return. See RabbitMQ.Client.Framing.*.Constants.
public	`string`	`ReplyText` (rw)	Human-readable text from the broker describing the reason for the return.
public	`string`	`RoutingKey` (rw)	The routing key used when the message was originally published.

Constructor Summary

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

Property Detail

public IBasicProperties BasicProperties (rw)

Summary

The content header of the message.

public byte[] Body (rw)

Summary

The message body.

public string Exchange (rw)

Summary

The exchange the returned message was originally published to.

public ushort ReplyCode (rw)

Summary

The AMQP reason code for the return. See RabbitMQ.Client.Framing.*.Constants.

public string ReplyText (rw)

Summary

Human-readable text from the broker describing the reason for the return.

public string RoutingKey (rw)

Summary

The routing key used when the message was originally published.

Constructor Detail

BasicReturnEventArgs

```
public BasicReturnEventArgs()
```

Summary

Default constructor.

Index | Namespace RabbitMQ.Client.Events

public delegate BasicReturnEventHandler

- extends `MulticastDelegate`

```
public delegate void BasicReturnEventHandler(IModel model, BasicReturnEventArgs args)
```

Return type	`void`	
	Name	**Type**
Parameters	model	IModel
	args	BasicReturnEventArgs

Summary

Delegate used to process Basic.Return events.
Index | Namespace RabbitMQ.Client.Events

public class CallbackExceptionEventArgs

- extends `EventArgs`

Summary

Describes an exception that was thrown during the library's invocation of an application-supplied callback handler.

Remarks

When an exception is thrown from a callback registered with part of the RabbitMQ .NET client library, it is caught, packaged into a CallbackExceptionEventArgs, and passed through the appropriate IModel's or IConnection's CallbackException event handlers. If an exception is thrown in a CallbackException handler, it is silently swallowed, as CallbackException is the last chance to handle these kinds of exception.

Code constructing CallbackExceptionEventArgs instances will usually place helpful information about the context of the call in the IDictionary available through the Detail property.

Property Summary

Flags	Type	Name	Summary
public	`IDictionary`	`Detail` (r)	Access helpful information about the context in which the wrapped exception was thrown.
public	`Exception`	`Exception` (r)	Access the wrapped exception.

Constructor Summary

Flags	Name	Summary
public	`CallbackExceptionEventArgs(Exception exception)`	Wrap an exception thrown by a callback.

Property Detail

public IDictionary Detail (r)

Summary

Access helpful information about the context in which the wrapped exception was thrown.

public Exception Exception (r)

Summary

Access the wrapped exception.

Constructor Detail

CallbackExceptionEventArgs

```
public CallbackExceptionEventArgs(Exception exception)
```

Parameters	Name	Type
	exception	`Exception`

Summary

Wrap an exception thrown by a callback.

Index | Namespace RabbitMQ.Client.Events

public delegate CallbackExceptionEventHandler

- extends `MulticastDelegate`

```
public delegate void CallbackExceptionEventHandler(object sender,
CallbackExceptionEventArgs e)
```

Return type	`void`	
	Name	**Type**
Parameters	sender	`object`
	e	`CallbackExceptionEventArgs`

Summary

Callback invoked when other callbacks throw unexpected exceptions.

Remarks

See also CallbackExceptionEventArgs.

Index | Namespace RabbitMQ.Client.Events

public delegate ConnectionShutdownEventHandler

- extends `MulticastDelegate`

```
public delegate void ConnectionShutdownEventHandler(IConnection connection,
ShutdownEventArgs reason)
```

Return type `void`

	Name	Type
Parameters	connection	IConnection
	reason	ShutdownEventArgs

Summary

Delegate used to process connection shutdown notifications.
Index | Namespace RabbitMQ.Client.Events

public class ConsumerEventArgs

- extends `EventArgs`

Summary

Event relating to a successful consumer registration or cancellation.

Property Summary

Flags	Type	Name	Summary
public	`string`	ConsumerTag (r)	Access the consumer-tag of the consumer the event relates to.

Constructor Summary

Flags	Name	Summary
public	ConsumerEventArgs(string consumerTag)	Construct an event containing the consumer-tag of the consumer the event relates to.

Property Detail

public string ConsumerTag (r)

Summary

Access the consumer-tag of the consumer the event relates to.

Constructor Detail

ConsumerEventArgs

```
public ConsumerEventArgs(string consumerTag)
```

Parameters	Name	Type
	consumerTag	`string`

Summary

Construct an event containing the consumer-tag of the consumer the event relates to.

Index | Namespace RabbitMQ.Client.Events

public delegate ConsumerEventHandler

- extends `MulticastDelegate`

```
public delegate void ConsumerEventHandler(object sender, ConsumerEventArgs e)
```

Return type	`void`	
	Name	**Type**
Parameters	sender	`object`
	e	ConsumerEventArgs

Summary

Callback for events relating to consumer registration and cancellation.
Index | Namespace RabbitMQ.Client.Events

public delegate ConsumerShutdownEventHandler

- extends `MulticastDelegate`

```
public delegate void ConsumerShutdownEventHandler(object sender, ShutdownEventArgs e)
```

Return type	`void`	
	Name	**Type**
Parameters	sender	`object`
	e	ShutdownEventArgs

Summary

Callback for events relating to consumer shutdown.

Remarks

Note that shutdown is different from cancellation: this delegate is invoked on IBasicConsumer's HandleModelShutdown method, not on the HandleBasicCancelOk method.

Index | Namespace RabbitMQ.Client.Events

public class EventingBasicConsumer

- extends DefaultBasicConsumer

Summary

Experimental class exposing an IBasicConsumer's methods as separate events.

Remarks

This class is experimental, and its interface may change radically from release to release.

Event Summary

Type	Name	Summary
BasicDeliverEventHandler	Received	Event fired on HandleBasicDeliver.
ConsumerEventHandler	Registered	Event fired on HandleBasicConsumeOk.
ConsumerShutdownEventHandler	Shutdown	Event fired on HandleModelShutdown.
ConsumerEventHandler	Unregistered	Event fired on HandleBasicCancelOk.

Constructor Summary

Flags	Name	Summary
public	EventingBasicConsumer()	(undocumented)

Method Summary

Flags	Name	Summary
public virtual	void HandleBasicCancelOk(string consumerTag)	Fires the Unregistered event.
public virtual	void HandleBasicConsumeOk(string consumerTag)	Fires the Registered event.
public virtual	void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool redelivered, string exchange, string routingKey, IBasicProperties properties, byte[] body)	Fires the Received event.
public virtual	void HandleModelShutdown(IModel model, ShutdownEventArgs reason)	Fires the Shutdown event.

Event Detail

BasicDeliverEventHandler Received

Summary

Event fired on HandleBasicDeliver.

ConsumerEventHandler Registered

Summary

Event fired on HandleBasicConsumeOk.

ConsumerShutdownEventHandler Shutdown

Summary

Event fired on HandleModelShutdown.

ConsumerEventHandler Unregistered

Summary

Event fired on HandleBasicCancelOk.

Constructor Detail

EventingBasicConsumer

```
public EventingBasicConsumer()
```

Method Detail

HandleBasicCancelOk

```
public virtual void HandleBasicCancelOk(string consumerTag)
```

Flags	public virtual	
Return type	`void`	
Parameters	**Name**	**Type**
	consumerTag	`string`

Summary

Fires the Unregistered event.

HandleBasicConsumeOk

```
public virtual void HandleBasicConsumeOk(string consumerTag)
```

Flags	public virtual	
Return type	`void`	
Parameters	**Name**	**Type**
	consumerTag	`string`

Summary

Fires the Registered event.

HandleBasicDeliver

```
public virtual void HandleBasicDeliver(string consumerTag, ulong deliveryTag, bool
redelivered, string exchange, string routingKey, IBasicProperties properties, byte[]
body)
```

Flags	public virtual	
Return type	`void`	
Parameters	**Name**	**Type**
	consumerTag	`string`
	deliveryTag	`ulong`
	redelivered	`bool`
	exchange	`string`
	routingKey	`string`
	properties	`IBasicProperties`
	body	`byte[]`

Summary

Fires the Received event.

HandleModelShutdown

```
public virtual void HandleModelShutdown(IModel model, ShutdownEventArgs reason)
```

Flags	public virtual	
Return type	`void`	
	Name	**Type**
Parameters	model	`IModel`
	reason	`ShutdownEventArgs`

Summary

Fires the Shutdown event.
Index | Namespace RabbitMQ.Client.Events

public class FlowControlEventArgs

- extends `EventArgs`

Summary

Event relating to flow control

Property Summary

Flags	Type	Name	Summary
public	`bool`	`Active` (r)	Access the flow control setting

Constructor Summary

Flags	Name	Summary
public	`FlowControlEventArgs(bool active)`	(undocumented)

Property Detail

public bool Active (r)

Summary

Access the flow control setting

Constructor Detail

FlowControlEventArgs

```
public FlowControlEventArgs(bool active)
```

Parameters	Name	Type
	active	`bool`

Index | Namespace RabbitMQ.Client.Events

public delegate FlowControlEventHandler

- extends `MulticastDelegate`

```
public delegate void FlowControlEventHandler(IModel sender, FlowControlEventArgs args)
```

Return type	`void`	
	Name	**Type**
Parameters	sender	IModel
	args	FlowControlEventArgs

Summary

Delegate used to process flow control events.
Index | Namespace RabbitMQ.Client.Events

public delegate ModelShutdownEventHandler

- extends `MulticastDelegate`

```
public delegate void ModelShutdownEventHandler(IModel model, ShutdownEventArgs reason)
```

Return type `void`

	Name	Type
Parameters	model	IModel
	reason	ShutdownEventArgs

Summary

Delegate used to process model shutdown notifications.

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.
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.Exceptions

public class BrokerUnreachableException

- extends `IOException`

Summary

Thrown when no connection could be opened during a ConnectionFactory.CreateConnection attempt.

Remarks

CreateConnection (optionally) handles redirections, so even a single-endpoint connection attempt may end up attempting to connect to multiple TCP endpoints. This exception contains information on how many times each endpoint was tried, and the outcome of the most recent attempt against each endpoint. See the ConnectionAttempts and ConnectionErrors properties.

Property Summary

Flags	Type	Name	Summary
public	`IDictionary`	`ConnectionAttempts` (r)	A map from AmqpTcpEndpoint to int, counting the number of attempts that were made against each endpoint.
public	`IDictionary`	`ConnectionErrors` (r)	A map from AmqpTcpEndpoint to Exception, recording the outcome of the most recent connection attempt against each endpoint.
public virtual	`IDictionary`	`Data` (r)	same as ConnectionErrors property

Constructor Summary

Flags	Name	Summary
public	`BrokerUnreachableException(IDictionary connectionAttempts, IDictionary connectionErrors, Exception Inner)`	Construct a BrokerUnreachableException. Expects maps as per the description of the ConnectionAttempts and ConnectionErrors properties. The inner exception is associated with only one connection attempt.

Method Summary

Flags	Name	Summary
public virtual	`string ToString()`	Provide a full description of the various connection attempts that were made, as well as the usual Exception stack trace.

Property Detail

public IDictionary ConnectionAttempts (r)

Summary

A map from AmqpTcpEndpoint to int, counting the number of attempts that were made against each endpoint.

public IDictionary ConnectionErrors (r)

Summary

A map from AmqpTcpEndpoint to Exception, recording the outcome of the most recent connection attempt against each endpoint.

public virtual IDictionary Data (r)

Summary

same as ConnectionErrors property

Constructor Detail

BrokerUnreachableException

```
public BrokerUnreachableException(IDictionary connectionAttempts, IDictionary
connectionErrors, Exception Inner)
```

	Name	Type
Parameters	connectionAttempts	IDictionary
	connectionErrors	IDictionary
	Inner	Exception

Summary

Construct a BrokerUnreachableException. Expects maps as per the description of the ConnectionAttempts and ConnectionErrors properties. The inner exception is associated with only one connection attempt.

Method Detail

ToString

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

Flags	public virtual
Return type	string

Summary

Provide a full description of the various connection attempts that were made, as well as the usual Exception stack trace.

Index | Namespace RabbitMQ.Client.Exceptions

public class ChannelAllocationException

- extends `Exception`

Summary

Thrown when a SessionManager cannot allocate a new channel number, or the requested channel number is already in use.

Property Summary

Flags	Type	Name	Summary
public	`int`	Channel (r)	Retrieves the channel number concerned; will return -1 in the case where "no more free channels" is being signalled, or a non-negative integer when "channel is in use" is being signalled.

Constructor Summary

Flags	Name	Summary
public	ChannelAllocationException(int channel)	Indicates that the specified channel is in use
public	ChannelAllocationException()	Indicates that there are no more free channels.

Property Detail

public int Channel (r)

Summary

Retrieves the channel number concerned; will return -1 in the case where "no more free channels" is being signalled, or a non-negative integer when "channel is in use" is being signalled.

Constructor Detail

ChannelAllocationException

```
public ChannelAllocationException(int channel)
```

Parameters	Name	Type
	channel	`int`

Summary

Indicates that the specified channel is in use

Param

The requested channel number

ChannelAllocationException

```
public ChannelAllocationException()
```

Summary

Indicates that there are no more free channels.

Index | Namespace RabbitMQ.Client.Exceptions

public class ConnectFailureException

- extends `SystemException`

Summary

Thrown when a connection to the broker fails

Constructor Summary

Flags	Name	Summary
public	`ConnectFailureException(string msg, Exception inner)`	(undocumented)

Constructor Detail

ConnectFailureException

```
public ConnectFailureException(string msg, Exception inner)
```

	Name	Type
Parameters	msg	`string`
	inner	`Exception`

Index | Namespace RabbitMQ.Client.Exceptions

public class OperationInterruptedException

- extends `Exception`

Summary

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.

Property Summary

Flags	Type	Name	Summary
public	ShutdownEventArgs	ShutdownReason (r)	Retrieves the explanation for the shutdown. May return null if no explanation is available.

Constructor Summary

Flags	Name	Summary
public	OperationInterruptedException(ShutdownEventArgs reason)	Construct an OperationInterruptedException with the passed-in explanation, if any.

Property Detail

public ShutdownEventArgs ShutdownReason (r)

Summary

Retrieves the explanation for the shutdown. May return null if no explanation is available.

Constructor Detail

OperationInterruptedException

```
public OperationInterruptedException(ShutdownEventArgs reason)
```

Parameters	Name	Type
	reason	ShutdownEventArgs

Summary

Construct an OperationInterruptedException with the passed-in explanation, if any.

Index | Namespace RabbitMQ.Client.Exceptions

public class PacketNotRecognizedException

- extends `ProtocolViolationException`

Summary

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.

Remarks

The peer's {'A','M','Q','P',txHi,txLo,major,minor} packet is decoded into instances of this class.

Property Summary

Flags	Type	Name	Summary
public	`int`	`ServerMajor` (r)	The peer's AMQP specification major version.
public	`int`	`ServerMinor` (r)	The peer's AMQP specification minor version.
public	`int`	`TransportHigh` (r)	The peer's high transport byte.
public	`int`	`TransportLow` (r)	The peer's low transport byte.

Constructor Summary

Flags	Name	Summary
public	`PacketNotRecognizedException(int transportHigh, int transportLow, int serverMajor, int serverMinor)`	Fills the new instance's properties with the values passed in.

Property Detail

public int ServerMajor (r)

Summary

The peer's AMQP specification major version.

public int ServerMinor (r)

Summary

The peer's AMQP specification minor version.

public int TransportHigh (r)

Summary

The peer's high transport byte.

public int TransportLow (r)

Summary

The peer's low transport byte.

Constructor Detail

PacketNotRecognizedException

```
public PacketNotRecognizedException(int transportHigh, int transportLow, int serverMajor,
int serverMinor)
```

	Name	Type
	transportHigh	`int`
Parameters	transportLow	`int`
	serverMajor	`int`
	serverMinor	`int`

Summary

Fills the new instance's properties with the values passed in.
Index | Namespace RabbitMQ.Client.Exceptions

public class PossibleAuthenticationFailureException

- extends `Exception`

Summary

Thrown when the likely cause is an authentication failure.

Constructor Summary

Flags	Name	Summary
public	PossibleAuthenticationFailureException(string msg, Exception inner)	(undocumented)

Constructor Detail

PossibleAuthenticationFailureException

```
public PossibleAuthenticationFailureException(string msg, Exception inner)
```

	Name	Type
Parameters	msg	`string`
	inner	`Exception`

Index | Namespace RabbitMQ.Client.Exceptions

public class ProtocolVersionMismatchException

- extends `ProtocolViolationException`

Summary

Thrown to indicate that the peer does not support the wire protocol version we requested immediately after opening the TCP socket.

Property Summary

Flags	Type	Name	Summary
public	`int`	`ClientMajor` (r)	The client's AMQP specification major version.
public	`int`	`ClientMinor` (r)	The client's AMQP specification minor version.
public	`int`	`ServerMajor` (r)	The peer's AMQP specification major version.
public	`int`	`ServerMinor` (r)	The peer's AMQP specification minor version.

Constructor Summary

Flags	Name	Summary
public	`ProtocolVersionMismatchException(int clientMajor, int clientMinor, int serverMajor, int serverMinor)`	Fills the new instance's properties with the values passed in.

Property Detail

public int ClientMajor (r)

Summary

The client's AMQP specification major version.

public int ClientMinor (r)

Summary

The client's AMQP specification minor version.

public int ServerMajor (r)

Summary

The peer's AMQP specification major version.

public int ServerMinor (r)

Summary

The peer's AMQP specification minor version.

Constructor Detail

ProtocolVersionMismatchException

```
public ProtocolVersionMismatchException(int clientMajor, int clientMinor, int serverMajor, int serverMinor)
```

	Name	Type
Parameters	clientMajor	`int`
	clientMinor	`int`
	serverMajor	`int`
	serverMinor	`int`

Summary

Fills the new instance's properties with the values passed in.
Index | Namespace RabbitMQ.Client.Exceptions

public class UnexpectedMethodException

- extends `Exception`

Summary

Thrown when the model receives an RPC reply that it wasn't expecting.

Property Summary

Flags	Type	Name	Summary
public	IMethod	Method (r)	The unexpected reply method.

Constructor Summary

Flags	Name	Summary
public	UnexpectedMethodException(IMethod method)	(undocumented)

Property Detail

public IMethod Method (r)

Summary

The unexpected reply method.

Constructor Detail

UnexpectedMethodException

```
public UnexpectedMethodException(IMethod method)
```

Parameters	Name	Type
	method	IMethod

Index | Namespace RabbitMQ.Client.Exceptions

public class UnsupportedMethodException

- extends `NotSupportedException`

Summary

Thrown when the model receives an RPC request it cannot satisfy.

Property Summary

Flags	Type	Name	Summary
public	`string`	MethodName (r)	The name of the RPC request that could not be sent.

Constructor Summary

Flags	Name	Summary
public	`UnsupportedMethodException(string methodName)`	(undocumented)

Property Detail

public string MethodName (r)

Summary

The name of the RPC request that could not be sent.

Constructor Detail

UnsupportedMethodException

```
public UnsupportedMethodException(string methodName)
```

Parameters	Name	Type
	methodName	`string`

Index | Namespace RabbitMQ.Client.Exceptions

public class UnsupportedMethodFieldException

- extends `NotSupportedException`

Summary

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.

Property Summary

Flags	Type	Name	Summary
public	`string`	`FieldName` (r)	The name of the unsupported field.
public	`string`	`MethodName` (r)	The name of the method involved.

Constructor Summary

Flags	Name	Summary
public	`UnsupportedMethodFieldException(string methodName, string fieldName)`	(undocumented)

Property Detail

public string FieldName (r)

Summary

The name of the unsupported field.

public string MethodName (r)

Summary

The name of the method involved.

Constructor Detail

UnsupportedMethodFieldException

```
public UnsupportedMethodFieldException(string methodName, string fieldName)
```

	Name	Type
Parameters	methodName	`string`
	fieldName	`string`

Index | Namespace RabbitMQ.Client.Exceptions

public class WireFormattingException

- extends `ProtocolViolationException`

Summary

Thrown when the wire-formatting code cannot encode a particular .NET value to AMQP protocol format.

Property Summary

Flags	Type	Name	Summary
public	`object`	`Offender` (r)	Object which this exception is complaining about; may be null if no particular offender exists

Constructor Summary

Flags	Name	Summary
public	`WireFormattingException(string message, object offender)`	Construct a WireFormattingException with the given offender
public	`WireFormattingException(string message)`	Construct a WireFormattingException with no particular offender (i.e. null)

Property Detail

public object Offender (r)

Summary

Object which this exception is complaining about; may be null if no particular offender exists

Constructor Detail

WireFormattingException

```
public WireFormattingException(string message, object offender)
```

	Name	Type
Parameters	message	`string`
	offender	`object`

Summary

Construct a WireFormattingException with the given offender

WireFormattingException

```
public WireFormattingException(string message)
```

	Name	Type
Parameters	message	`string`

Summary

Construct a WireFormattingException with no particular offender (i.e. null)

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.Client.MessagePatterns

public class SimpleRpcServer

- implements `IDisposable`

Summary

Implements a simple RPC service, responding to requests received via a Subscription.

Remarks

This class interprets requests such as those sent by instances of SimpleRpcClient.

The basic pattern for implementing a service is to subclass SimpleRpcServer, overriding HandleCall and HandleCast as appropriate, and then to create a Subscription object for receiving requests from clients, and start an instance of the SimpleRpcServer subclass with the Subscription.

```
string queueName = "ServiceRequestQueue"; // See also Subscription ctors
using (IConnection conn = new ConnectionFactory()
                                  .CreateConnection(serverAddress)) {
    using (IModel ch = conn.CreateModel()) {
        Subscription sub = new Subscription(ch, queueName);
        new MySimpleRpcServerSubclass(sub).MainLoop();
    }
}
```

Note that this class itself does not declare any resources (exchanges, queues or bindings). The Subscription we use for receiving RPC requests should have already declared all the resources we need. See the Subscription constructors and the Subscription.Bind method.

If you are implementing a service that responds to "jms/stream-message"-formatted requests (as implemented by RabbitMQ.Client.Content.IStreamMessageReader), override HandleStreamMessageCall. Otherwise, override HandleSimpleCall or HandleCall as appropriate. Asynchronous, one-way requests are dealt with by HandleCast etc.

Every time a request is successfully received and processed within the server's MainLoop, the request message is Ack()ed using Subscription.Ack before the next request is retrieved. This causes the Subscription object to take care of acknowledging receipt and processing of the request message.

If transactional service is enabled, via SetTransactional(), then after every successful ProcessRequest, IModel.TxCommit is called. Making use of transactional service has effects on all parts of the application that share an IModel instance, completely changing the style of interaction with the AMQP server. For this reason, it is initially disabled, and must be explicitly enabled with a call to SetTransactional(). Please see the documentation for SetTransactional() for details.

To stop a running RPC server, call Close(). This will in turn Close() the Subscription, which will cause MainLoop() to return to its caller.

Unless overridden, ProcessRequest examines properties in the request content header, and uses them to dispatch to one of the Handle[...]() methods. See the documentation for ProcessRequest and each Handle[...] method for details.

See

- `RabbitMQ.Client.MessagePatterns.SimpleRpcClient`

Property Summary

Flags	Type	Name	Summary
public	`bool`	`Transactional` (r)	Returns true if we are in "transactional" mode, or false if we are not.

Constructor Summary

Flags	Name	Summary
public	SimpleRpcServer(Subscription subscription)	Create, but do not start, an instance that will receive requests via the given Subscription.

Method Summary

Flags	Name	Summary
public	void Close()	Shut down the server, causing MainLoop() to return to its caller.
public virtual	byte[] HandleCall(bool isRedelivered, IBasicProperties requestProperties, byte[] body, out IBasicProperties replyProperties)	Called by ProcessRequest(), this is the most general method that handles RPC-style requests.
public virtual	void HandleCast(bool isRedelivered, IBasicProperties requestProperties, byte[] body)	Called by ProcessRequest(), this is the most general method that handles asynchronous, one-way requests.
public virtual	byte[] HandleSimpleCall(bool isRedelivered, IBasicProperties requestProperties, byte[] body, out IBasicProperties replyProperties)	Called by the default HandleCall() implementation as a fallback.
public virtual	void HandleSimpleCast(bool isRedelivered, IBasicProperties requestProperties, byte[] body)	Called by the default HandleCast() implementation as a fallback.
public virtual	void HandleStreamMessageCall(IStreamMessageBuilder replyWriter, bool isRedelivered, IBasicProperties requestProperties, object[] args)	Called by HandleCall and HandleCast when a "jms/stream-message" request is received.
public	void MainLoop()	Enters the main loop of the RPC service.
public virtual	void ProcessRequest(BasicDeliverEventArgs evt)	Process a single request received from our subscription.
public	void SetTransactional()	Enables transactional mode.

Property Detail

public bool Transactional (r)

Summary

Returns true if we are in "transactional" mode, or false if we are not.

Constructor Detail

SimpleRpcServer

```
public SimpleRpcServer(Subscription subscription)
```

Parameters	Name	Type
	subscription	Subscription

Summary

Create, but do not start, an instance that will receive requests via the given Subscription.

Remarks

The instance is initially in non-transactional mode. See SetTransactional().

Call MainLoop() to start the request-processing loop.

Method Detail

Close

```
public void Close()
```

Flags	public
Return type	void

Summary

Shut down the server, causing MainLoop() to return to its caller.

Remarks

Acts by calling Close() on the server's Subscription object.

HandleCall

```
public virtual byte[] HandleCall(bool isRedelivered, IBasicProperties requestProperties,
byte[] body, out IBasicProperties replyProperties)
```

Flags	public virtual
Return type	byte[]

	Name	Type
Parameters	isRedelivered	bool
	requestProperties	IBasicProperties
	body	byte[]
	replyProperties	out IBasicProperties

Summary

Called by ProcessRequest(), this is the most general method that handles RPC-style requests.

Remarks

This method should map requestProperties and body to replyProperties and the returned byte array.

The default implementation checks requestProperties.ContentType, and if it is "jms/stream-message" (i.e. the current value of StreamMessageBuilder.MimeType), parses it using StreamMessageReader and delegates to HandleStreamMessageCall before encoding and returning the reply. If the ContentType is any other value, the request is passed to HandleSimpleCall instead.

The isRedelivered flag is true when the server knows for sure that it has tried to send this request previously (although not necessarily to this application). It is not a reliable indicator of previous receipt, however - the only claim it makes is that a delivery attempt was made, not that the attempt succeeded. Be careful if you choose to use the isRedelivered flag.

HandleCast

```
public virtual void HandleCast(bool isRedelivered, IBasicProperties requestProperties,
byte[] body)
```

Flags	public virtual
Return type	`void`

	Name	Type
Parameters	isRedelivered	`bool`
	requestProperties	`IBasicProperties`
	body	`byte[]`

Summary

Called by ProcessRequest(), this is the most general method that handles asynchronous, one-way requests.

Remarks

The default implementation checks requestProperties.ContentType, and if it is "jms/stream-message" (i.e. the current value of StreamMessageBuilder.MimeType), parses it using StreamMessageReader and delegates to HandleStreamMessageCall, passing in null as the replyWriter parameter to indicate that no reply is desired or possible. If the ContentType is any other value, the request is passed to HandleSimpleCast instead.

The isRedelivered flag is true when the server knows for sure that it has tried to send this request previously (although not necessarily to this application). It is not a reliable indicator of previous receipt, however - the only claim it makes is that a delivery attempt was made, not that the attempt succeeded. Be careful if you choose to use the isRedelivered flag.

HandleSimpleCall

```
public virtual byte[] HandleSimpleCall(bool isRedelivered, IBasicProperties
requestProperties, byte[] body, out IBasicProperties replyProperties)
```

Flags	public virtual
Return type	`byte[]`

	Name	Type
Parameters	isRedelivered	`bool`
	requestProperties	`IBasicProperties`
	body	`byte[]`
	replyProperties	`out IBasicProperties`

Summary

Called by the default HandleCall() implementation as a fallback.

Remarks

If the MIME ContentType of the request did not match any of the types specially recognised (e.g. "jms/stream-message"), this method is called instead with the raw bytes of the request. It should fill in replyProperties (or set it to null) and return a byte array to send back to the remote caller as a reply message.

HandleSimpleCast

```
public virtual void HandleSimpleCast(bool isRedelivered, IBasicProperties
requestProperties, byte[] body)
```

Flags	public virtual
Return type	`void`

	Name	Type
Parameters	isRedelivered	`bool`
	requestProperties	`IBasicProperties`
	body	`byte[]`

Summary

Called by the default HandleCast() implementation as a fallback.

Remarks

If the MIME ContentType of the request did not match any of the types specially recognised (e.g. "jms/stream-message"), this method is called instead with the raw bytes of the request.

HandleStreamMessageCall

```
public virtual void HandleStreamMessageCall(IStreamMessageBuilder replyWriter, bool
isRedelivered, IBasicProperties requestProperties, object[] args)
```

Flags	public virtual	
Return type	`void`	
	Name	**Type**
Parameters	replyWriter	`IStreamMessageBuilder`
	isRedelivered	`bool`
	requestProperties	`IBasicProperties`
	args	`object[]`

Summary

Called by HandleCall and HandleCast when a "jms/stream-message" request is received.

Remarks

The args array contains the values decoded by HandleCall or HandleCast.

The replyWriter parameter will be null if we were called from HandleCast, in which case a reply is not expected or possible, or non-null if we were called from HandleCall. Use the methods of replyWriter in this case to assemble your reply, which will be sent back to the remote caller.

This default implementation does nothing, which effectively sends back an empty reply to any and all remote callers.

MainLoop

```
public void MainLoop()
```

Flags	public
Return type	`void`

Summary

Enters the main loop of the RPC service.

Remarks

Retrieves requests repeatedly from the service's subscription. Each request is passed to ProcessRequest. Once ProcessRequest returns, the request is acknowledged via Subscription.Ack(). If transactional mode is enabled, TxCommit is then called. Finally, the loop begins again.

Runs until the subscription ends, which happens either as a result of disconnection, or of a call to Close().

ProcessRequest

```
public virtual void ProcessRequest(BasicDeliverEventArgs evt)
```

Flags	public virtual
Return type	void

Parameters	**Name**	**Type**
	evt	BasicDeliverEventArgs

Summary

Process a single request received from our subscription.

Remarks

If the request's properties contain a non-null, non-empty CorrelationId string (see IBasicProperties), it is assumed to be a two-way call, requiring a response. The ReplyTo header property is used as the reply address (via PublicationAddress.Parse, unless that fails, in which case it is treated as a simple queue name), and the request is passed to HandleCall().

If the CorrelationId is absent or empty, the request is treated as one-way asynchronous event, and is passed to HandleCast().

Usually, overriding HandleCall(), HandleCast(), or one of their delegates is sufficient to implement a service, but in some cases overriding ProcessRequest() is required. Overriding ProcessRequest() gives the opportunity to implement schemes for detecting interaction patterns other than simple request/response or one-way communication.

SetTransactional

```
public void SetTransactional()
```

Flags	public
Return type	void

Summary

Enables transactional mode.

Remarks

Once enabled, transactional mode is not only enabled for all users of the underlying IModel instance, but cannot be disabled without shutting down the entire IModel (which involves shutting down all the services depending on it, and should not be undertaken lightly).

This method calls IModel.TxSelect, every time it is called. (TxSelect is idempotent, so this is harmless.)

Index | Namespace RabbitMQ.Client.MessagePatterns

public class Subscription

- implements `IDisposable`
- implements `IEnumerable`
- implements `IEnumerator`

Summary

Manages a subscription to a queue or exchange.

Remarks

This convenience class abstracts away from much of the detail involved in receiving messages from a queue or an exchange.

Once created, the Subscription consumes from a queue (using a QueueingBasicConsumer). Received deliveries can be retrieved by calling Next(), or by using the Subscription as an IEnumerator in, for example, a foreach loop.

Note that if the "noAck" option is enabled (which it is by default), then received deliveries are automatically acked within the server before they are even transmitted across the network to us. Calling Ack() on received events will always do the right thing: if "noAck" is enabled, nothing is done on an Ack() call, and if "noAck" is disabled, IModel.BasicAck() is called with the correct parameters.

Property Summary

Flags	Type	Name	Summary
public	`IBasicConsumer`	`Consumer` (r)	Retrieve the IBasicConsumer that is receiving the messages from the server for us. Normally, you will not need to access this property - use Next() and friends instead.
public	`string`	`ConsumerTag` (r)	Retrieve the consumer-tag that this subscription is using. Will usually be a server-generated name.
public	`BasicDeliverEventArgs`	`LatestEvent` (r)	Returns the most recent value returned by Next(), or null when either no values have been retrieved yet, the end of the subscription has been reached, or the most recent value has already been Ack()ed. See also the documentation for Ack().
public	`IModel`	`Model` (r)	Retrieve the IModel our subscription is carried by.
public	`bool`	`NoAck` (r)	Returns true if we are in "noAck" mode, where calls to Ack() will be no-ops, and where the server acks messages before they are delivered to us. Returns false if we are in a mode where calls to Ack() are required, and where such calls will actually send an acknowledgement message across the network to the server.
public	`string`	`QueueName` (r)	Retrieve the queue name we have subscribed to.

Constructor Summary

Flags	Name	Summary
public	`Subscription(IModel model, string queueName, bool noAck)`	Creates a new Subscription, with full control over both "noAck" mode and the name of the queue.
public	`Subscription(IModel model, string queueName)`	Creates a new Subscription in "noAck" mode, consuming from a named queue.

Method Summary

Flags	Name	Summary
public	`void Ack()`	

		If LatestEvent is non-null, passes it to Ack(BasicDeliverEventArgs). Causes LatestEvent to become null.
public	void Ack(BasicDeliverEventArgs evt)	If we are not in "noAck" mode, calls IModel.BasicAck with the delivery-tag from the passed in event; otherwise, sends nothing to the server. In both cases, if the passed-in event is the same as LatestEvent (by pointer comparison), sets LatestEvent to null.
public	void Close()	Closes this Subscription, cancelling the consumer record in the server.
public	bool Next(int millisecondsTimeout, out BasicDeliverEventArgs result)	Retrieves the next incoming delivery in our subscription queue, or times out after a specified number of milliseconds.
public	BasicDeliverEventArgs Next()	Retrieves the next incoming delivery in our subscription queue.

Property Detail

public IBasicConsumer Consumer (r)

Summary

Retrieve the IBasicConsumer that is receiving the messages from the server for us. Normally, you will not need to access this property - use Next() and friends instead.

public string ConsumerTag (r)

Summary

Retrieve the consumer-tag that this subscription is using. Will usually be a server-generated name.

public BasicDeliverEventArgs LatestEvent (r)

Summary

Returns the most recent value returned by Next(), or null when either no values have been retrieved yet, the end of the subscription has been reached, or the most recent value has already been Ack()ed. See also the documentation for Ack().

public IModel Model (r)

Summary

Retrieve the IModel our subscription is carried by.

public bool NoAck (r)

Summary

Returns true if we are in "noAck" mode, where calls to Ack() will be no-ops, and where the server acks messages before they are delivered to us. Returns false if we are in a mode where calls to Ack() are required, and where such calls will actually send an acknowledgement message across the network to the server.

public string QueueName (r)

Summary

Retrieve the queue name we have subscribed to.

Constructor Detail

Subscription

```
public Subscription(IModel model, string queueName, bool noAck)
```

	Name	Type
Parameters	model	IModel
	queueName	string
	noAck	bool

Summary

Creates a new Subscription, with full control over both "noAck" mode and the name of the queue.

Subscription

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

	Name	Type
Parameters	model	IModel
	queueName	string

Summary

Creates a new Subscription in "noAck" mode, consuming from a named queue.

Method Detail

Ack

```
public void Ack()
```

Flags	public
Return type	void

Summary

If LatestEvent is non-null, passes it to Ack(BasicDeliverEventArgs). Causes LatestEvent to become null.

Ack

```
public void Ack(BasicDeliverEventArgs evt)
```

Flags	public	
Return type	void	
Parameters	**Name**	**Type**
	evt	BasicDeliverEventArgs

Summary

If we are not in "noAck" mode, calls IModel.BasicAck with the delivery-tag from the passed in event; otherwise, sends nothing to the server. In both cases, if the passed-in event is the same as LatestEvent (by pointer comparison), sets LatestEvent to null.

Remarks

Make sure that this method is only called with events that originated from this Subscription - other usage will have unpredictable results.

Close

```
public void Close()
```

Flags	public
Return type	void

Summary

Closes this Subscription, cancelling the consumer record in the server.

Next

```
public bool Next(int millisecondsTimeout, out BasicDeliverEventArgs result)
```

Flags	public
Return type	bool

	Name	Type
Parameters	millisecondsTimeout	int
	result	out BasicDeliverEventArgs

Summary

Retrieves the next incoming delivery in our subscription queue, or times out after a specified number of milliseconds.

Remarks

Returns false only if the timeout expires before either a delivery appears or the end-of-stream is reached. If false is returned, the out parameter "result" is set to null, but LatestEvent is not updated.

Returns true to indicate a delivery or the end-of-stream.

If a delivery is already waiting in the queue, or one arrives before the timeout expires, it is removed from the queue and placed in the "result" out parameter. If the end-of-stream is detected before the timeout expires, "result" is set to null.

Whenever this method returns true, it updates LatestEvent to the value placed in "result" before returning.

End-of-stream can arise through the action of the Subscription.Close() method, or through the closure of the IModel or its underlying IConnection.

This method does not acknowledge any deliveries at all (but in "noAck" mode, the server will have auto-acknowledged each event before it is even sent across the wire to us).

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 an item or the end of the stream to become available. Usage of such a timeout is equivalent to calling Next() with no arguments (modulo predictable method signature differences).

Next

```
public BasicDeliverEventArgs Next()
```

Flags	public
Return type	BasicDeliverEventArgs

Summary

Retrieves the next incoming delivery in our subscription queue.

Remarks

Returns null when the end of the stream is reached and on every subsequent call. End-of-stream can arise through the action of the Subscription.Close() method, or through the closure of the IModel or its underlying IConnection.

Updates LatestEvent to the value returned.

Does not acknowledge any deliveries at all (but in "noAck" mode, the server will have auto-acknowledged each event before it is even sent across the wire to us).

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.
SharedQueueEnumerator	Implementation of the IEnumerator interface, for permitting SharedQueue to be used in foreach loops.
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

Index | Namespace RabbitMQ.Util

public class DebugUtil

Summary

Miscellaneous debugging and development utilities.

Remarks

Not part of the public API.

Method Summary

Flags	Name	Summary
public static	void Dump(byte[] bytes, TextWriter writer)	Print a hex dump of the supplied bytes to the supplied TextWriter.
public static	void Dump(byte[] bytes)	Print a hex dump of the supplied bytes to stdout.
public static	void DumpKeyValue(string key, object value, TextWriter writer, int indent)	Prints an indented key/value pair; used by DumpProperties()
public static	void DumpProperties(object value, TextWriter writer, int indent)	Dump properties of objects to the supplied writer.

Method Detail

Dump

```
public static void Dump(byte[] bytes, TextWriter writer)
```

Flags	public static	
Return type	void	
Parameters	**Name**	**Type**
	bytes	byte[]
	writer	TextWriter

Summary

Print a hex dump of the supplied bytes to the supplied TextWriter.

Dump

```
public static void Dump(byte[] bytes)
```

Flags	public static	
Return type	void	
Parameters	**Name**	**Type**
	bytes	byte[]

Summary

Print a hex dump of the supplied bytes to stdout.

DumpKeyValue

```
public static void DumpKeyValue(string key, object value, TextWriter writer, int indent)
```

Flags	public static
Return type	void

	Name	Type
Parameters	key	`string`
	value	`object`
	writer	`TextWriter`
	indent	`int`

Summary

Prints an indented key/value pair; used by DumpProperties()

Remarks

Recurses into the value using DumpProperties().

DumpProperties

```
public static void DumpProperties(object value, TextWriter writer, int indent)
```

	Name	Type
Flags	public static	
Return type	`void`	
Parameters	value	`object`
	writer	`TextWriter`
	indent	`int`

Summary

Dump properties of objects to the supplied writer.

Index | Namespace RabbitMQ.Util

public class Either

Summary

Models the disjoint union of two alternatives, a "left" alternative and a "right" alternative.

Remarks

Borrowed from ML, Haskell etc.

Property Summary

Flags	Type	Name	Summary
public	EitherAlternative	Alternative (r)	Retrieve the alternative represented by this instance.
public	object	Value (r)	Retrieve the value carried by this instance.

Method Summary

Flags	Name	Summary
public static	Either Left(object value)	Constructs an Either instance representing a Left alternative.
public static	Either Right(object value)	Constructs an Either instance representing a Right alternative.

Property Detail

public EitherAlternative Alternative (r)

Summary

Retrieve the alternative represented by this instance.

public object Value (r)

Summary

Retrieve the value carried by this instance.

Method Detail

Left

```
public static Either Left(object value)
```

Flags	public static
Return type	Either

Parameters	Name	Type
	value	object

Summary

Constructs an Either instance representing a Left alternative.

Right

```
public static Either Right(object value)
```

Flags	public static
Return type	Either

Parameters	Name	Type
	value	object

Summary

Constructs an Either instance representing a Right alternative.
Index | Namespace RabbitMQ.Util

public enum struct EitherAlternative

- extends Enum

Summary

Used internally by class Either.

Field Summary

Flags	Type	Name	Summary
public const	EitherAlternative	Left	(undocumented)
public const	EitherAlternative	Right	(undocumented)

Field Detail

public const EitherAlternative Left

public const EitherAlternative Right

Index | Namespace RabbitMQ.Util

public class IntAllocator

Nested types: IntervalList

Constructor Summary

Flags	Name	Summary
public	IntAllocator(int start, int end)	(undocumented)

Method Summary

Flags	Name	Summary
public	int Allocate()	(undocumented)
public	void Free(int id)	(undocumented)
public	bool Reserve(int id)	(undocumented)

Constructor Detail

IntAllocator

```
public IntAllocator(int start, int end)
```

	Name	Type
Parameters	start	int
	end	int

Method Detail

Allocate

```
public int Allocate()
```

Flags	public
Return type	int

Free

```
public void Free(int id)
```

Flags	public	
Return type	void	
Parameters	**Name**	**Type**
	id	int

Reserve

```
public bool Reserve(int id)
```

Flags	public	
Return type	bool	
Parameters	**Name**	**Type**
	id	int

Index | Namespace RabbitMQ.Util

class IntervalList

- declared within IntAllocator

Field Summary

Flags	Type	Name	Summary
public	int	End	(undocumented)
public	IntAllocator.IntervalList	Next	(undocumented)
public	int	Start	(undocumented)

Constructor Summary

Flags	Name	Summary
public	IntervalList(int start, int end)	(undocumented)

Method Summary

Flags	Name	Summary
public static	IntAllocator.IntervalList FromArray(int[] xs, int length)	(undocumented)
public static	IntAllocator.IntervalList Merge(IntAllocator.IntervalList x, IntAllocator.IntervalList y)	(undocumented)

Field Detail

public int End

public IntAllocator.IntervalList Next

public int Start

Constructor Detail

IntervalList

```
public IntervalList(int start, int end)
```

	Name	Type
Parameters	start	int
	end	int

Method Detail

FromArray

```
public static IntAllocator.IntervalList FromArray(int[] xs, int length)
```

Flags	public static
Return type	IntAllocator.IntervalList

	Name	Type
Parameters	xs	`int[]`
	length	`int`

Merge

```
public static IntAllocator.IntervalList Merge(IntAllocator.IntervalList x,
IntAllocator.IntervalList y)
```

Flags	public static
Return type	IntAllocator.IntervalList

	Name	Type
Parameters	x	IntAllocator.IntervalList
	y	IntAllocator.IntervalList

Index | Namespace RabbitMQ.Util

public class IntAllocator

Nested types: IntervalList

Constructor Summary

Flags	Name	Summary
public	IntAllocator(int start, int end)	(undocumented)

Method Summary

Flags	Name	Summary
public	int Allocate()	(undocumented)
public	void Free(int id)	(undocumented)
public	bool Reserve(int id)	(undocumented)

Constructor Detail

IntAllocator

```
public IntAllocator(int start, int end)
```

	Name	Type
Parameters	start	int
	end	int

Method Detail

Allocate

```
public int Allocate()
```

Flags	public
Return type	int

Free

```
public void Free(int id)
```

Flags	public	
Return type	void	
Parameters	**Name**	**Type**
	id	int

Reserve

```
public bool Reserve(int id)
```

Flags	public	
Return type	bool	
Parameters	**Name**	**Type**
	id	int

Index | Namespace RabbitMQ.Util

public class NetworkBinaryReader

- extends `BinaryReader`

Summary

Subclass of BinaryReader that reads integers etc in correct network order.

Remarks

Kludge to compensate for .NET's broken little-endian-only BinaryReader. Relies on BinaryReader always being little-endian.

Constructor Summary

Flags	Name	Summary
public	`NetworkBinaryReader(Stream input, Encoding encoding)`	Construct a NetworkBinaryReader over the given input stream, reading strings using the given encoding.
public	`NetworkBinaryReader(Stream input)`	Construct a NetworkBinaryReader over the given input stream.

Method Summary

Flags	Name	Summary
public virtual	`double ReadDouble()`	Override BinaryReader's method for network-order.
public virtual	`short ReadInt16()`	Override BinaryReader's method for network-order.
public virtual	`int ReadInt32()`	Override BinaryReader's method for network-order.
public virtual	`long ReadInt64()`	Override BinaryReader's method for network-order.
public virtual	`single ReadSingle()`	Override BinaryReader's method for network-order.
public virtual	`ushort ReadUInt16()`	Override BinaryReader's method for network-order.
public virtual	`uint ReadUInt32()`	Override BinaryReader's method for network-order.
public virtual	`ulong ReadUInt64()`	Override BinaryReader's method for network-order.
public static	`BinaryReader TemporaryBinaryReader(byte[] bytes)`	Helper method for constructing a temporary BinaryReader over a byte[].

Constructor Detail

NetworkBinaryReader

```
public NetworkBinaryReader(Stream input, Encoding encoding)
```

	Name	Type
Parameters	input	`Stream`
	encoding	`Encoding`

Summary

Construct a NetworkBinaryReader over the given input stream, reading strings using the given encoding.

NetworkBinaryReader

```
public NetworkBinaryReader(Stream input)
```

Parameters	Name	Type
	input	Stream

Summary

Construct a NetworkBinaryReader over the given input stream.

Method Detail

ReadDouble

```
public virtual double ReadDouble()
```

Flags public virtual

Return type double

Summary

Override BinaryReader's method for network-order.

ReadInt16

```
public virtual short ReadInt16()
```

Flags public virtual

Return type short

Summary

Override BinaryReader's method for network-order.

ReadInt32

```
public virtual int ReadInt32()
```

Flags public virtual

Return type int

Summary

Override BinaryReader's method for network-order.

ReadInt64

```
public virtual long ReadInt64()
```

Flags public virtual

Return type long

Summary

Override BinaryReader's method for network-order.

ReadSingle

```
public virtual single ReadSingle()
```

Flags public virtual

Return type `single`

Summary

Override BinaryReader's method for network-order.

ReadUInt16

```
public virtual ushort ReadUInt16()
```

Flags public virtual

Return type `ushort`

Summary

Override BinaryReader's method for network-order.

ReadUInt32

```
public virtual uint ReadUInt32()
```

Flags public virtual

Return type `uint`

Summary

Override BinaryReader's method for network-order.

ReadUInt64

```
public virtual ulong ReadUInt64()
```

Flags public virtual

Return type `ulong`

Summary

Override BinaryReader's method for network-order.

TemporaryBinaryReader

```
public static BinaryReader TemporaryBinaryReader(byte[] bytes)
```

Flags public static

Return type `BinaryReader`

Parameters

Name	Type
bytes	`byte[]`

Summary

Helper method for constructing a temporary BinaryReader over a byte[].

Index | Namespace RabbitMQ.Util

public class NetworkBinaryWriter

- extends `BinaryWriter`

Summary

Subclass of BinaryWriter that writes integers etc in correct network order.

Remarks

Kludge to compensate for .NET's broken little-endian-only BinaryWriter.

See also NetworkBinaryReader.

Constructor Summary

Flags	Name	Summary
public	`NetworkBinaryWriter(Stream output, Encoding encoding)`	Construct a NetworkBinaryWriter over the given input stream, reading strings using the given encoding.
public	`NetworkBinaryWriter(Stream output)`	Construct a NetworkBinaryWriter over the given input stream.

Method Summary

Flags	Name	Summary
public static	`BinaryWriter TemporaryBinaryWriter(int initialSize)`	Helper method for constructing a temporary BinaryWriter streaming into a fresh MemoryStream provisioned with the given initialSize.
public static	`byte[] TemporaryContents(BinaryWriter w)`	Helper method for extracting the byte[] contents of a BinaryWriter over a MemoryStream, such as constructed by TemporaryBinaryWriter.
public virtual	`void Write(single f)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(double d)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(short i)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(ulong i)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(int i)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(ushort i)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(long i)`	Override BinaryWriter's method for network-order.
public virtual	`void Write(uint i)`	Override BinaryWriter's method for network-order.

Constructor Detail

NetworkBinaryWriter

```
public NetworkBinaryWriter(Stream output, Encoding encoding)
```

	Name	Type
Parameters	output	`Stream`
	encoding	`Encoding`

Summary

Construct a NetworkBinaryWriter over the given input stream, reading strings using the given encoding.

NetworkBinaryWriter

```
public NetworkBinaryWriter(Stream output)
```

	Name	Type
Parameters	output	`Stream`

Summary

Construct a NetworkBinaryWriter over the given input stream.

Method Detail

TemporaryBinaryWriter

```
public static BinaryWriter TemporaryBinaryWriter(int initialSize)
```

Flags	public static
Return type	`BinaryWriter`

	Name	Type
Parameters	initialSize	`int`

Summary

Helper method for constructing a temporary BinaryWriter streaming into a fresh MemoryStream provisioned with the given initialSize.

TemporaryContents

```
public static byte[] TemporaryContents(BinaryWriter w)
```

Flags	public static
Return type	`byte[]`

	Name	Type
Parameters	w	`BinaryWriter`

Summary

Helper method for extracting the byte[] contents of a BinaryWriter over a MemoryStream, such as constructed by TemporaryBinaryWriter.

Write

```
public virtual void Write(single f)
```

Flags	public virtual
Return type	`void`

	Name	Type
Parameters	f	`single`

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(double d)
```

Flags	public virtual
Return type	void

Parameters	**Name**	**Type**
	d	double

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(short i)
```

Flags	public virtual
Return type	void

Parameters	**Name**	**Type**
	i	short

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(ulong i)
```

Flags	public virtual
Return type	void

Parameters	**Name**	**Type**
	i	ulong

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(int i)
```

Flags	public virtual
Return type	void

Parameters	**Name**	**Type**
	i	int

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(ushort i)
```

Flags	public virtual
Return type	`void`

Parameters	**Name**	**Type**
	i	`ushort`

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(long i)
```

Flags	public virtual
Return type	`void`

Parameters	**Name**	**Type**
	i	`long`

Summary

Override BinaryWriter's method for network-order.

Write

```
public virtual void Write(uint i)
```

Flags	public virtual
Return type	`void`

Parameters	**Name**	**Type**
	i	`uint`

Summary

Override BinaryWriter's method for network-order.
Index | Namespace RabbitMQ.Util

public class SharedQueue

- implements `IEnumerable`

Summary

A thread-safe shared queue implementation.

Constructor Summary

Flags	Name	Summary
public	SharedQueue()	Construct a fresh, empty SharedQueue.

Method Summary

Flags	Name	Summary
public	void Close()	Close the queue. Causes all further Enqueue() operations to throw EndOfStreamException, and all pending or subsequent Dequeue() operations to throw an EndOfStreamException once the queue is empty.
public	bool Dequeue(int millisecondsTimeout, out object result)	Retrieve the first item from the queue, or return nothing if no items are available after the given timeout
public	object Dequeue()	Retrieve the first item from the queue, or block if none available
public	object DequeueNoWait(object defaultValue)	Retrieve the first item from the queue, or return defaultValue immediately if no items are available
public	void Enqueue(object o)	Place an item at the end of the queue.

Constructor Detail

SharedQueue

`public SharedQueue()`

Summary

Construct a fresh, empty SharedQueue.

Method Detail

Close

`public void Close()`

Flags public

Return type `void`

Summary

Close the queue. Causes all further Enqueue() operations to throw EndOfStreamException, and all pending or subsequent Dequeue() operations to throw an EndOfStreamException once the queue is empty.

Dequeue

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

Flags public

Return type `bool`

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

Summary

Retrieve the first item from the queue, or return nothing if no items are available after the given timeout
Remarks

If one or more items are present on the queue at the time the call is made, the call will return immediately. Otherwise, the calling thread blocks until either an item appears on the queue, or millisecondsTimeout milliseconds have elapsed.

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

If no items were 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 an item to become available. Usage of such a timeout is equivalent to calling Dequeue() with no arguments. See also the MSDN documentation for System.Threading.Monitor.Wait(object,int).

If no items are present and the queue is in a closed state, or if at any time while waiting the queue transitions to a closed state (by a call to Close()), this method will throw EndOfStreamException.

Dequeue

```
public object Dequeue()
```

Flags	public
Return type	`object`

Summary

Retrieve the first item from the queue, or block if none available
Remarks

Callers of Dequeue() will block if no items are available until some other thread calls Enqueue() or the queue is closed. In the latter case this method will throw EndOfStreamException.

DequeueNoWait

```
public object DequeueNoWait(object defaultValue)
```

Flags	public
Return type	`object`

	Name	Type
Parameters	defaultValue	`object`

Summary

Retrieve the first item from the queue, or return defaultValue immediately if no items are available
Remarks

If one or more objects are present in the queue at the time of the call, the first item is removed from the queue and returned. Otherwise, the defaultValue that was passed in is returned immediately. This defaultValue may be null, or in cases where null is part of the range of the queue, may be some other sentinel object. The difference between DequeueNoWait() and Dequeue() is that DequeueNoWait() will not block when no items are available in the queue, whereas Dequeue() will.

If at the time of call the queue is empty and in a closed state (following a call to Close()), then this method will throw EndOfStreamException.

Enqueue

```
public void Enqueue(object o)
```

Flags	public
Return type	void

Parameters	**Name**	**Type**
	o	object

Summary

Place an item at the end of the queue.

Remarks

If there is a thread waiting for an item to arrive, the waiting thread will be woken, and the newly Enqueued item will be passed to it. If the queue is closed on entry to this method, EndOfStreamException will be thrown.

Index | Namespace RabbitMQ.Util

public class SharedQueueEnumerator

- implements `IEnumerator`

Summary

Implementation of the IEnumerator interface, for permitting SharedQueue to be used in foreach loops.

Constructor Summary

Flags	Name	Summary
public	SharedQueueEnumerator(SharedQueue queue)	Construct an enumerator for the given SharedQueue.

Constructor Detail

SharedQueueEnumerator

```
public SharedQueueEnumerator(SharedQueue queue)
```

Parameters	Name	Type
	queue	SharedQueue

Summary

Construct an enumerator for the given SharedQueue.

Index | Namespace RabbitMQ.Util

public class XmlUtil

Summary

Miscellaneous helpful XML utilities.

Method Summary

Flags	Name	Summary
public static	XmlTextWriter CreateIndentedXmlWriter(Stream stream)	Constructs an indenting XmlTextWriter that writes to the supplied stream.
public static	XmlTextWriter CreateIndentedXmlWriter(string path)	Constructs an indenting XmlTextWriter that writes to the supplied file name.
public static	XmlTextWriter CreateIndentedXmlWriter()	Constructs an indenting XmlTextWriter that writes to a fresh MemoryStream.
public static	XmlDocument SerializeObject(Type serializationType, object obj)	Serializes an arbitrary serializable object to an XML document.

Method Detail

CreateIndentedXmlWriter

```
public static XmlTextWriter CreateIndentedXmlWriter(Stream stream)
```

Flags	public static
Return type	XmlTextWriter

Parameters	Name	Type
	stream	Stream

Summary

Constructs an indenting XmlTextWriter that writes to the supplied stream.

CreateIndentedXmlWriter

```
public static XmlTextWriter CreateIndentedXmlWriter(string path)
```

Flags	public static
Return type	XmlTextWriter

Parameters	Name	Type
	path	string

Summary

Constructs an indenting XmlTextWriter that writes to the supplied file name.

CreateIndentedXmlWriter

```
public static XmlTextWriter CreateIndentedXmlWriter()
```

Flags	public static
Return type	XmlTextWriter

Summary

Constructs an indenting XmlTextWriter that writes to a fresh MemoryStream.

SerializeObject

```
public static XmlDocument SerializeObject(Type serializationType, object obj)
```

Flags	public static	
Return type	`XmlDocument`	
	Name	**Type**
Parameters	serializationType	`Type`
	obj	`object`

Summary

Serializes an arbitrary serializable object to an XML document.