

Contents

Contents	3
Chapter 1:Preface	11
Audience	11
Overview of the Guide	11
Introduction to FMQ C#RTL	11
Typographical Conventions	12
Related Documentation	12
Chapter 2:Datatypes and Constants	15
Naming convention	15
Chapter 3:Error Handling	17
Chapter 4:Function Reference	19
Helpers	20
CsHashTable	20
Constructor	20
Put	20
Get	20
RemoveElement	21
HashTableEnumerator	21
hasMoreElemets	21
nextKeyElement	22
nextValueElement	22
JMS Interfaces	23
CsByteMessage	23
getBodyLength	23
readBoolean	23
readByte	24
readBytes	24
readChar	25
readDouble	25
readFloat	26
readInt	26
readLong	26
readShort	27
readUnsignedByte	27
readUnsignedShort	28
readUTF	28
reset	29
writeBoolean	29
writeByte	29
writeBytes	30
writeChar	30

writeDouble	31
writeFloat	31
writeInt	32
writeLong	32
writeShort	33
writeUTF	33
CsConnection	34
CsConnectionConsumer	34
CsConnectionFactory	34
CsConnectionMetaData	34
CsDestination	35
CsException	35
checkForException	35
CJMSException	35
Constructor	35
Constructor	36
checkForException	36
printStackTrace	37
getErrorCode	37
getLinkedException	37
CsExceptionListener	37
onException	38
CsJMSException	38
Constructor	38
Constructor	38
checkForException	39
printStackTrace	39
getErrorCode	39
getLinkedException	40
CsMapMessage	40
getBoolean	40
getByte	41
getChar	41
getDouble	42
getFloat	42
getInt	43
getLong	43
getMapNames	44
getShort	44
getString	45
itemExists	45
setBoolean	46
setByte	46
setBytes	47
setBytes	47
setChar	48
setDouble	48
setFloat	49

setInt	49
setLong	50
setShort	50
setString	51
CsMessage	51
acknowledge	51
clearBody	52
clearProperties	52
getBooleanProperty	53
getByteProperty	53
getDoubleProperty	54
getFloatProperty	54
getIntProperty	55
getJMSCorrelationID	55
getJMSCorrelationIDAsBytes	56
getJMSDeliveryMode	56
getJMSDestination	57
getJMSExpiration	57
getJMSMessageID	58
getJMSPriority	59
getJMSRedelivered	59
getJMSReplyTo	60
getJMSTimestamp	60
getJMSType	61
getLongProperty	61
getObjectProperty	62
getShortProperty	62
getStringProperty	63
propertyExists	63
setBooleanProperty	64
setByteProperty	64
setDoubleProperty	65
setFloatProperty	65
setIntProperty	66
setJMSCorrelationID	66
setJMSDeliveryMode	67
setJMSDestination	67
setJMSExpiration	68
setJMSMessageID	68
setJMSPriority	69
setJMSRedelivered	69
setJMSReplyTo	70
setJMSTimestamp	70
setJMSType	71
setLongProperty	71
setObjectProperty	72
setshortProperty	72
setStringProperty	73

CsMessageConsumer	73
CsMessageListener	73
CsServerSession	74
CsServerSessionPool	74
CsSession	74
CsStreamMessage	75
readBoolean	75
readByte	76
readBytes	76
readChar	77
readDouble	77
readFloat	77
readInt	78
readLong	78
readShort	79
readString	79
reset	80
writeBoolean	80
writeByte	80
writeBytes	81
writeBytes	81
writeChar	82
writeDouble	82
writeFloat	83
writeInt	83
writeLong	84
writeShort	84
writeString	84
CsTextMessage	85
getText	85
setText	86
Naming and Lookup (JNDI)	87
CsInitialContext	87
Constructor	87
Lookup	87
PTP	89
CsQueue	89
getQueueName	89
toString	89
CsQueueConnection	90
createQueueSession	90
close	91
getClientID	91
setClientID	92
start	92
stop	93
getExceptionListener	93

setExceptionListener	94
CsQueueConnectionFactory	94
createQueueConnection	94
createQueueConnection	95
CsQueueReceiver	95
getQueue	96
close	96
getMessageListener	97
getMessageSelector	97
receive	97
recieve	98
reiveNoWait	98
set	99
CsQueueRequestor	99
close	100
request.	100
request.	101
QueueBrowser	101
close	101
getMessageSelector	102
getQueue	102
CsQueueSender	103
getQueue	103
close	103
getDeliveryMode	104
getDestination	104
getDisableMessageID	104
getDisableMessageTimestamp	105
getPriority	105
getTimeToLive	106
send	106
send	107
send	107
send	108
setDeliveryMode	108
setDisableMessageID	109
setDisableMessageTimeStamp	109
setPriority	110
setTimeToLive	110
CsQueueSession	111
close	111
commit	111
createBrowser	112
createBrowser	112
createBytesMessage	113
createMapMessage	113
createQueue	114
createStreamMessage	114

createTemporaryQueue	115
createTextMessage	115
createTextMessage	115
getMessageListener	116
recover	116
rollback	117
run	117
setMessageListener	117
CsTemporaryQueue	118
remove	118
Publish/Subscribe	119
CsTemporaryTopic	119
remove	119
CsTopic	119
getTopicName	120
toString	120
CsTopicConnection	120
createTopicSession	121
close	121
getClientID	122
setClientID	122
start	122
stop	123
getExceptionListener	123
setExceptionListener	124
CsTopicConnectionFactory	124
createTopicConnection	124
createTopicConnection	125
CsTopicPublisher	125
getTopic	126
publish	126
publish	126
publish	127
publish	128
close	128
getDeliveryMode	129
getDestination	129
getDisableMessageID	130
getDisableMessageTimestamp	130
getPriority	130
getTimeToLive	131
send	131
send	132
send	132
send	133
setDeliveryMode	134
setDisableMessageID	134
setDisableMessageTimestamp	135

setPriority	135
setTimeToLive	136
CsTopicRequestor	136
close	136
request	137
request	137
CsTopicSession	138
createPublisher	138
createSubscriber	139
createSubscriber	139
close	140
commit	140
createBytesMessage	140
createDurableSubscriber	141
createDurableSubscriber	141
createMapMessage	142
createStreamMessage	143
createTemporaryQueue	143
createTextMessage	143
createTextMessage	144
createTopic	144
getMessageListener	145
recover	145
rollback	145
setMessageListener	146
unsubscribe	146
CsTopicSubscriber	147
close	147
getMessageListener	147
getMessageSelector	148
receive	148
receive	149
receiveNoWait	149
setMessageListener	150
getNoLocal	150
getTopic	151
Chapter 5:Using the Samples Programs	153
Organization of Samples Provided	153
Compiling and Running Samples	153
Limitations of C#RTL	154
Chapter 6:Frequently Asked Questions	155

Preface

Audience

This guide is designed to assist developers working on the .Net technology to understand and use FioranoMQ C#RTL library. The developer must have a fair knowledge of the C# programming language.

Overview of the Guide


This guide contains the following contents.

- An introduction to FioranoMQ C# RunTime library.
- Detailed description of the APIs included in this library.
- Details on compiling and running C#RTL sample applications.
- Frequently Asked Questions (FAQ).

Introduction to FMQ C#RTL

The C#RTL allows C# applications to communicate seamlessly with C, C++ and java programs. This library helps in bridging the gap between J2EE and .NET domain and enables .NET applications to directly talk to the Java server. This version of C#RTL supports secure and non-secure TCP and HTTP connections on Win32 platform for Point-to-Point and Publish/Subscribe communication models.

The C#RTL has been designed to provide maximum conformance to JMS specifications. All public APIs have similar signature as the corresponding Java APIs specified by JMS. All classes have a similar naming convention. Exception handling is also similar to that specified by JMS with all APIs throwing a `CJMSException` with a specific error Code and/or error description.

 For more information on JMS specifications, please refer to Java Message Service Specification on the Sun Microsystems website.

Typographical Conventions

To locate and interpret information easily, the manual uses consistent visual cues, and standard text formats. The Table below lists the various conventions used in the manual.




Conventions	Description
 Note	Additional/important information.
 See Also	Reference to important information located elsewhere in the documentation framework.
 Warning	Information regarding a potential aberration if a component is used incorrectly.
1. Numbered List	Instructions that must be followed in a sequential order.
■ Bulleted List	A list where sequence is immaterial.
C:\Programs\FioranoMQ	Directory paths
start()	Class names, methods, and interfaces.
<code>public class ChatService</code> <code>extends FioranoBISService</code>	Code samples
<code>fiorano.FioranoBIS.common.*</code>	This style is used to highlight: <ol style="list-style-type: none"> 1. Program code within paragraphs 2. Values to be entered by users in a procedure

TABLE 1 Typographical Conventions

Related Documentation

For complete information on the available RunTime libraries, we strongly recommend that you go through the entire range of FioranoMQ RTL Documentation.

Document Name	Description
C Runtime Library Guide	Contains the description of the FioranoMQ C Runtime APIs.
Native C ++ Runtime Library Guide	Contains a description of the FioranoMQ native C++ Runtime APIs.

Document Name	Description
JNI-based C ++ Runtime Library Guide	Provides an introduction to FioranoMQ JNI based C++ Runtime APIs
JavaDocs	Contains "javadoc" style documentation on all public Fiorano classes.

TABLE 2 Related Documentation

Datatypes and Constants

This chapter contains an overview of the CRTL specific data types and constants. A brief explanation of each data type along with sizes, is provided.

Naming convention

The C#RTL adheres to the following naming convention for you to easily identify the classes, constants and member functions with the corresponding definitions in JMS specifications.

Type	Naming Convention	Example
Class	Cs<JMSSClass name>	<code>CsTopicPublisher(JMS:TopicPublisher)</code>
Constant	same as in JMS spec	<code>NON_PERSISTENT</code>
Function	same as in JMS spec	<code>publish</code>

Error Handling

3

The C#RTL uses exceptions to provide error handling. In event of an error in the C#RTL layer, `CJMSEException` with a specific Error Code and description is thrown.

The Exception can be caught at the application level and the associated message can be read using the public api `getMessage()`. The exception handling is also exhaustive in that it provides the complete function stack trace at the moment of the error.

The exception stack is maintained in the Thread Local Storage, so that the exception that occurred in one thread doesn't interfere the flow of other threads.

The stack trace can be printed on the console using the API call `printStackTrace()`.

```
try
{
//Application code
}
catch(CJMSEException *e)
{
cout << e->getMessage();
e->printStackTrace();
}
```



For more information, read the `CJMSEException` section.
