



# QAnywhere™ User's Guide

Part number: DC20051-01-0902-01  
Last modified: October 2004

---

Copyright © 1989–2004 Sybase, Inc. Portions copyright © 2001–2004 iAnywhere Solutions, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of iAnywhere Solutions, Inc. iAnywhere Solutions, Inc. is a subsidiary of Sybase, Inc.

Sybase, SYBASE (logo), AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, AnswerBase, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, ASEP, AvantGo, AvantGo Application Alerts, AvantGo Mobile Delivery, AvantGo Mobile Document Viewer, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BayCam, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client Services, Client-Library, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, Dynamo, e-ADK, E-Anywhere, e-Biz Integrator, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, E-Whatever, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Channel, M-Business Network, M-Business Server, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, ML Query, MobicATS, My AvantGo, My AvantGo Media Channel, My AvantGo Mobile Marketing, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASis, OASis logo, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Business Interchange, Open Client, Open Client/Server, Open Client/Server Interfaces, Open ClientConnect, Open Gateway, Open ServerConnect, Open Solutions, Optima++, Orchestration Studio, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power Through Knowledge, power.stop, Power++, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, Powersoft Portfolio, Powersoft Professional, PowerStage, PowerStudio, PowerTips, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, QAnywhere, Rapport, Relational Beans, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report Workbench, Report-Execute, Resource Manager, RW-DisplayLib, RW-Library, S.W.I.F.T. Message Format Libraries, SAFE, SAFE/PRO, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILLS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL Server SNMP SubAgent, SQL Server/CFT, SQL Server/DBM, SQL SMART, SQL Station, SQL Toolset, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase User Workbench, Sybase Virtual Server Architecture, SybaseWare, Syber Financial, SyberAssist, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Versacore, Viewer, VisualWriter, VQL, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, WarehouseArchitect, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, and XP Server are trademarks of Sybase, Inc. or its subsidiaries.

Certicom, MobileTrust, and SSL Plus are trademarks and Security Builder is a registered trademark of Certicom Corp. Copyright © 1997–2001 Certicom Corp. Portions are Copyright © 1997–1998, Consensus Development Corporation, a wholly owned subsidiary of Certicom Corp. All rights reserved. Contains an implementation of NR signatures, licensed under U.S. patent 5,600,725. Protected by U.S. patents 5,787,028; 4,745,568; 5,761,305. Patents pending.

All other trademarks are property of their respective owners.

---

# Contents

<b>About This Manual</b>	<b>vii</b>
SQL Anywhere Studio documentation . . . . .	viii
Documentation conventions . . . . .	xi
Finding out more and providing feedback . . . . .	xiii
<b>1 Introduction to QAnywhere</b>	<b>1</b>
Application-to-application messaging . . . . .	2
What QAnywhere does . . . . .	3
QAnywhere architecture . . . . .	5
Quick start . . . . .	10
<b>2 Tutorial: A Sample QAnywhere Application</b>	<b>11</b>
About the tutorial . . . . .	12
Lesson 1: Start MobiLink with messaging . . . . .	13
Lesson 2: Create a client message store . . . . .	16
Lesson 3: Run the TestMessage application . . . . .	18
Lesson 4: Send a message . . . . .	20
Lesson 5: Explore the TestMessage client source code . . . . .	22
Lesson 6: Start a JMS connector . . . . .	27
Tutorial cleanup . . . . .	29
<b>3 Setting Up QAnywhere Messaging</b>	<b>31</b>
Setting up server-side components . . . . .	32
Setting up client-side components . . . . .	35
Using push notifications . . . . .	40
Using JMS Connectors . . . . .	42
Using QAnywhere messaging and MobiLink data synchro- nization together . . . . .	51
Setting up a failover mechanism . . . . .	53
<b>4 Writing QAnywhere Client Applications</b>	<b>55</b>
Introduction . . . . .	56
Overview of writing a client application . . . . .	58
Understanding QAnywhere message addresses . . . . .	59
Initializing the QAnywhere client API . . . . .	60
Setting QAManager properties . . . . .	64
Sending QAnywhere messages . . . . .	67
Receiving messages synchronously . . . . .	69

---

Receiving messages asynchronously . . . . .	70
Reading very large messages . . . . .	72
Handling push notifications and network status changes . . . . .	73
Implementing transactional messaging . . . . .	75
Shutting down QAnywhere . . . . .	77
Deploying QAnywhere applications . . . . .	78
<b>5 QAnywhere Agent</b>	<b>79</b>
QAnywhere Agent syntax . . . . .	80
<b>6 Writing Secure Messaging Applications</b>	<b>95</b>
Creating a secure client message store . . . . .	96
Encrypting the communication stream . . . . .	98
Using password authentication with MobiLink . . . . .	99
<b>7 QAnywhere Transmission Rules</b>	<b>101</b>
Transmission rules . . . . .	102
Schedule syntax . . . . .	105
Transmission rule variables . . . . .	110
Delete rules . . . . .	118
<b>8 QAnywhere C++ API Reference</b>	<b>121</b>
Class AcknowledgementMode . . . . .	122
Class MessageProperties . . . . .	123
Class MessageType . . . . .	126
Class QABinaryMessage . . . . .	127
Class QAEError . . . . .	135
Class QAManager . . . . .	138
Class QAManagerBase . . . . .	141
Class QAManagerFactory . . . . .	153
Class QAMessage . . . . .	155
Class QAMessageListener . . . . .	167
Class QATextMessage . . . . .	168
Class QATransactionalManager . . . . .	171
<b>9 iAnywhere.QAnywhere.Client namespace</b>	<b>173</b>
AcknowledgementMode enumeration . . . . .	174
MessageProperties class . . . . .	175
MessageType enumeration . . . . .	180
QABinaryMessage class . . . . .	181
QAEException class . . . . .	192
QAManager class . . . . .	195
QAManagerBase class . . . . .	201
QAManagerBase.MessageListener delegate . . . . .	219

---

QAManagerFactory class . . . . .	220
QAMessage class . . . . .	224
QAPropertyType enumeration . . . . .	237
QATextMessage class . . . . .	238
QATransactionalManager class . . . . .	242

<b>Index</b>	<b>249</b>
--------------	------------



---

# About This Manual

Subject	This manual describes QAnywhere, which defines a messaging platform for mobile and wireless clients as well as traditional desktop and laptop clients.
Audience	This manual is for users of Adaptive Server Anywhere and other relational database systems who want to add messaging to their mobile applications, or who want to build new mobile application-to-application messaging solutions.

---

# SQL Anywhere Studio documentation

## The SQL Anywhere Studio documentation

This book is part of the SQL Anywhere documentation set. This section describes the books in the documentation set and how you can use them.

The SQL Anywhere Studio documentation is available in a variety of forms: in an online form that combines all books in one large help file; as separate PDF files for each book; and as printed books that you can purchase. The documentation consists of the following books:

- ◆ **Introducing SQL Anywhere Studio** This book provides an overview of the SQL Anywhere Studio database management and synchronization technologies. It includes tutorials to introduce you to each of the pieces that make up SQL Anywhere Studio.
- ◆ **What's New in SQL Anywhere Studio** This book is for users of previous versions of the software. It lists new features in this and previous releases of the product and describes upgrade procedures.
- ◆ **Adaptive Server Anywhere Database Administration Guide** This book covers material related to running, managing, and configuring databases and database servers.
- ◆ **Adaptive Server Anywhere SQL User's Guide** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.
- ◆ **Adaptive Server Anywhere SQL Reference Manual** This book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.
- ◆ **Adaptive Server Anywhere Programming Guide** This book describes how to build and deploy database applications using the C, C++, and Java programming languages. Users of tools such as Visual Basic and PowerBuilder can use the programming interfaces provided by those tools. It also describes the Adaptive Server Anywhere ADO.NET data provider.
- ◆ **Adaptive Server Anywhere SNMP Extension Agent User's Guide** This book describes how to configure the Adaptive Server Anywhere SNMP Extension Agent for use with SNMP management applications to manage Adaptive Server Anywhere databases.
- ◆ **Adaptive Server Anywhere Error Messages** This book provides a complete listing of Adaptive Server Anywhere error messages together with diagnostic information.



- 
- ◆ **SQL Anywhere Studio Security Guide** This book provides information about security features in Adaptive Server Anywhere databases. Adaptive Server Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. This book may be of interest to those who wish to run the current version of Adaptive Server Anywhere in a manner equivalent to the C2-certified environment.
  - ◆ **MobiLink Administration Guide** This book describes how to use the MobiLink data synchronization system for mobile computing, which enables sharing of data between a single Oracle, Sybase, Microsoft or IBM database and many Adaptive Server Anywhere or UltraLite databases.
  - ◆ **MobiLink Clients** This book describes how to set up and synchronize Adaptive Server Anywhere and UltraLite remote databases.
  - ◆ **MobiLink Tutorials** This book provides several tutorials to help you learn MobiLink technology.
  - ◆ **MobiLink Server-Initiated Synchronization User's Guide** This book describes MobiLink server-initiated synchronization, a feature of MobiLink that allows you to initiate synchronization from the consolidated database.
  - ◆ **QAnywhere User's Guide** This manual describes MobiLink QAnywhere, a messaging platform that enables the development and deployment of messaging applications for mobile and wireless clients, as well as traditional desktop and laptop clients.
  - ◆ **iAnywhere Solutions ODBC Drivers** This book describes how to set up ODBC drivers to access consolidated databases other than Adaptive Server Anywhere from the MobiLink synchronization server and from Adaptive Server Anywhere remote data access.
  - ◆ **SQL Remote User's Guide** This book describes all aspects of the SQL Remote data replication system for mobile computing, which enables sharing of data between a single Adaptive Server Anywhere or Adaptive Server Enterprise database and many Adaptive Server Anywhere databases using an indirect link such as e-mail or file transfer.
  - ◆ **SQL Anywhere Studio Help** This book includes the context-sensitive help for Sybase Central, Interactive SQL, and other graphical tools. It is not included in the printed documentation set.
  - ◆ **UltraLite Database User's Guide** This book is intended for all UltraLite developers. It introduces the UltraLite database system and provides information common to all UltraLite programming interfaces.

- 
- ◆ **UltraLite Interface Guides** A separate book is provided for each UltraLite programming interface. Some of these interfaces are provided as UltraLite components for rapid application development, and others are provided as static interfaces for C, C++, and Java development.

In addition to this documentation set, PowerDesigner and InfoMaker include their own online documentation.

Documentation formats SQL Anywhere Studio provides documentation in the following formats:

- ◆ **Online documentation** The online documentation contains the complete SQL Anywhere Studio documentation, including both the books and the context-sensitive help for SQL Anywhere tools. The online documentation is updated with each maintenance release of the product, and is the most complete and up-to-date source of documentation.

To access the online documentation on Windows operating systems, choose Start ► Programs ► SQL Anywhere 9 ► Online Books. You can navigate the online documentation using the HTML Help table of contents, index, and search facility in the left pane, as well as using the links and menus in the right pane.

To access the online documentation on UNIX operating systems, see the HTML documentation under your SQL Anywhere installation.

- ◆ **PDF books** The SQL Anywhere books are provided as a set of PDF files, viewable with Adobe Acrobat Reader.

The PDF books are accessible from the online books, or from the Windows Start menu.

- ◆ **Printed books** The complete set of books is available from Sybase sales or from eShop, the Sybase online store, at <http://eshop.sybase.com/eshop/documentation>.

---

# Documentation conventions

This section lists the typographic and graphical conventions used in this documentation.

## Syntax conventions

The following conventions are used in the SQL syntax descriptions:

- ◆ **Keywords** All SQL keywords appear in upper case, like the words ALTER TABLE in the following example:

**ALTER TABLE** [ *owner*.]*table-name*

- ◆ **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown like the words *owner* and *table-name* in the following example:

**ALTER TABLE** [ *owner*.]*table-name*

- ◆ **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots), like *column-constraint* in the following example:

**ADD** *column-definition* [ *column-constraint*, . . . ]

One or more list elements are allowed. In this example, if more than one is specified, they must be separated by commas.

- ◆ **Optional portions** Optional portions of a statement are enclosed by square brackets.

**RELEASE SAVEPOINT** [ *savepoint-name* ]

These square brackets indicate that the *savepoint-name* is optional. The square brackets should not be typed.

- ◆ **Options** When none or only one of a list of items can be chosen, vertical bars separate the items and the list is enclosed in square brackets.

[ **ASC** | **DESC** ]

For example, you can choose one of ASC, DESC, or neither. The square brackets should not be typed.

- ◆ **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces and a bar is used to separate the options.

[ **QUOTES** { **ON** | **OFF** } ]

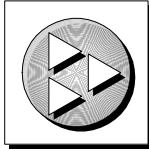
If the QUOTES option is used, one of ON or OFF must be provided. The brackets and braces should not be typed.

---

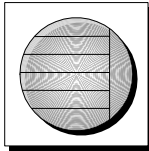
## Graphic icons

The following icons are used in this documentation.

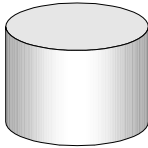
- ◆ A client application.



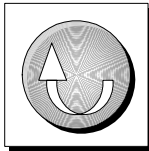
- ◆ A database server, such as Sybase Adaptive Server Anywhere.



- ◆ A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.



- ◆ Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink Synchronization Server and the SQL Remote Message Agent.



- ◆ A programming interface.



---

## Finding out more and providing feedback

### Finding out more

Additional information and resources, including a code exchange, are available at the iAnywhere Developer Network at <http://www.ianywhere.com/developer/>.

If you have questions or need help, you can post messages to the iAnywhere Solutions newsgroups listed below.

When you write to one of these newsgroups, always provide detailed information about your problem, including the build number of your version of SQL Anywhere Studio. You can find this information by typing **dbeng9 -v** at a command prompt.

The newsgroups are located on the *forums.sybase.com* news server. The newsgroups include the following:

- ◆ [sybase.public.sqlanywhere.general](#)
- ◆ [sybase.public.sqlanywhere.linux](#)
- ◆ [sybase.public.sqlanywhere.mobilink](#)
- ◆ [sybase.public.sqlanywhere.product\\_futures\\_discussion](#)
- ◆ [sybase.public.sqlanywhere.replication](#)
- ◆ [sybase.public.sqlanywhere.ultralite](#)
- ◆ [ianywhere.public.sqlanywhere.qanywhere](#)

#### **Newsgroup disclaimer**

iAnywhere Solutions has no obligation to provide solutions, information or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and ensure its operation and availability.

iAnywhere Solutions Technical Advisors as well as other staff assist on the newsgroup service when they have time available. They offer their help on a volunteer basis and may not be available on a regular basis to provide solutions and information. Their ability to help is based on their workload.

### Feedback

We would like to receive your opinions, suggestions, and feedback on this documentation.

You can e-mail comments and suggestions to the SQL Anywhere documentation team at [iasdoc@ianywhere.com](mailto:iasdoc@ianywhere.com). Although we do not reply to e-mails sent to that address, we read all suggestions with interest.

---

In addition, you can provide feedback on the documentation and the software through the newsgroups listed above.

---

## CHAPTER 1

# Introduction to QAnywhere

About this chapter

QAnywhere is a comprehensive application-to-application messaging system for mobile users. It provides the infrastructure for you to write applications that exchange messages with remote applications located on a variety of devices running on Windows or Windows CE operating systems.

Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Application-to-application messaging</a>	2
<a href="#">What QAnywhere does</a>	3
<a href="#">QAnywhere architecture</a>	5
<a href="#">Quick start</a>	10

---

## Application-to-application messaging

Application-to-application messaging, including mobile device to mobile device and mobile device to and from the enterprise, permits communication between custom programs running on mobile or wireless devices and a centrally located server application. Messaging is a useful application-to-application communication mechanism in a variety of situations:

- ◆ It provides communication in occasionally-connected environments.

The store-and-forward nature of messaging means that messages can be constructed even when the destination application is not reachable over the network; the message is delivered when the network becomes available.

QAnywhere messages are exchanged via a central server, so that the sender and recipient of a message never have to be connected to the network at the same time.

- ◆ It provides network-independent communication.

QAnywhere messages can be transported over TCP/IP, HTTP, or HTTPS protocols. They can also be delivered from a Windows CE handheld device by ActiveSync. The message itself is independent of the network protocol, and can be received by an application that communicates over a different network.

QAnywhere handles the challenges of wireless networks, such as slow speed, spotty geographic coverage, and dropped network connections. It can protect proprietary or sensitive information by encrypting all messages sent over public networks. You can customize the delivery of messages using transmission rules so that, for example, messages are transmitted at the most convenient times.

QAnywhere compresses and, optionally, encrypts data sent between mobile applications and enterprise servers. Furthermore, it implements a store-and-forward messaging paradigm that guarantees message delivery.

QAnywhere is designed for messaging solutions on a variety of handheld devices. This system provides both a QAnywhere C++ API and a QAnywhere .NET API to provide solutions to developers with different skill sets.

QAnywhere permits seamless communication with other messaging systems that have a JMS interface. This allows integration with J2EE applications.



## What QAnywhere does

QAnywhere provides the following application-to-application features and components.

- ◆ **Programming API** The object-oriented QAnywhere API provides the infrastructure to build messaging applications for Windows desktop and Windows CE devices.
- ◆ **Store-and-forward** QAnywhere applications store messages in queues until a connection between the client and the server is available for data transmission.

- ◆ **Complements data synchronization** QAnywhere applications use relational databases as a temporary message store. The relational database ensures that the message store has security, transaction-based computing, and the other benefits of relational databases.

The use of Adaptive Server Anywhere relational databases as message stores makes it easy to use QAnywhere together with a data synchronization solution. Both use MobiLink synchronization as the underlying mechanism for exchanging information between client and server.

- ◆ **Integration with external messaging systems** In addition to exchanging messages among QAnywhere applications, you can integrate QAnywhere clients into external messaging systems that support a JMS interface.
- ◆ **Encryption** Messages can be sent encrypted using a 128-bit encryption key. In addition, messages stores can be encrypted using the AES algorithm.
- ◆ **Compression** Messages can be stored compressed using the L277 (deflation variant) algorithm. Doing so reduces inflation of compressed data that is sometimes seen with the more common LZW algorithm.
- ◆ **Authentication** Users can be authenticated using an existing authentication service provided by another application in your organization.
- ◆ **Multiple networks** QAnywhere works over any wired or wireless network that supports TCP/IP or HTTP.
- ◆ **Failover** You can run multiple MobiLink synchronization servers so that there are alternate servers in case one fails.

- 
- ◆ **Multiple queues** Support for multiple arbitrarily-named queues on client devices permits multiple client applications to coexist on a single device. Applications can send and receive on any number of queues. Messages can be sent between applications that are coexisting on the same device and between applications on different devices.
  - ◆ **Server-initiated send and receive** QAnywhere can push messages to client devices, allowing client applications to implement message-driven logic.
  - ◆ **Rules for managing the message store** You can create rules that specify when message transmission should occur, as well as customize the persistence of messages in the message stores.
  - ◆ **Resumable downloads** Large messages or groups of messages are sent to QAnywhere clients in piecemeal fashion to minimize the retransmission of data during network failures.
  - ◆ **Guaranteed delivery** QAnywhere guarantees the delivery of messages once and only once.

## QAnywhere architecture

This section explains the architecture of QAnywhere messaging applications. The discussion begins with a simple messaging scenario and then progresses to more advanced scenarios.

Client applications send and receive messages using the QAnywhere programming API. Messages are queued in the client message store. Message transmission is the exchange of messages between client message stores through a central QAnywhere server message store.

Following are typical messaging scenarios that are supported by QAnywhere:

- ◆ **Simple messaging** For exchanging messages among QAnywhere clients. Client applications control when to transmit messages between the client and server message stores.  
☞ See “Simple messaging scenario” on page 5.
- ◆ **Messaging with push notifications** For exchanging messages among QAnywhere clients. In this scenario, the QAnywhere server can initiate message transmission between client and server message stores.  
☞ See “Scenario for messaging with push notifications” on page 6.
- ◆ **Messaging with external messaging systems** For exchanging messages among QAnywhere clients or an external system that supplies a JMS provider, such as BEA WebLogic or Sybase EAServer.  
☞ See “Scenario for messaging with external messaging systems” on page 8.

Push notifications and external messaging systems can be used together, providing the most general solution.

### Simple messaging scenario

A simple QAnywhere messaging setup is illustrated in the following diagram. For simplicity, only a single client is shown.

Client  
message  
store

This setup includes the following components:

- ◆ **Server message store** At the server, the messages are stored in a relational database. The database must be set up as a MobiLink consolidated database, and may be any supported consolidated database

---

(Adaptive Server Anywhere, Adaptive Server Enterprise, Microsoft SQL Server, DB2, or Oracle).

- ◆ **Client message store** The messages at each client are also stored in a relational database. The database used is Adaptive Server Anywhere.
- ◆ **MobiLink synchronization server with messaging** MobiLink synchronization provides the transport for transmitting and tracking messages between QAnywhere clients and the server. MobiLink provides security, authentication, encryption, and flexibility. It also allows messaging to be combined with data synchronization.

The MobiLink synchronization server must be started with messaging enabled in order to manage QAnywhere message transmission. You do this by supplying the MobiLink synchronization server -m command line option.

☞ For more information, see [“Starting the MobiLink synchronization server for QAnywhere messaging” on page 33](#).

- ◆ **QAnywhere Agent** The QAnywhere Agent manages transmitting messages on the client side.

☞ For more information, see [“Running the QAnywhere Agent” on page 37](#).

- ◆ **QAnywhere client application** An application written using the QAnywhere C++ API or the QAnywhere .NET API makes function calls to send and receive messages.

☞ For information about writing applications using the QAnywhere API, see [“Writing QAnywhere Client Applications” on page 55](#).

Messages are sent and received by the QAnywhere clients. Messages at the server will not be picked up until the client initiates a message transmission. QAnywhere clients use **policies** to determine when to carry out a message transmission. Policies include on-demand, automatic, scheduled, and custom. The on-demand policy permits the user to control message transmission. The automatic policy initiates a message transmission each time a message at the client is ready for delivery.

☞ For more information, see [“Determining when message transmission should occur on the client” on page 37](#).

## Scenario for messaging with push notifications

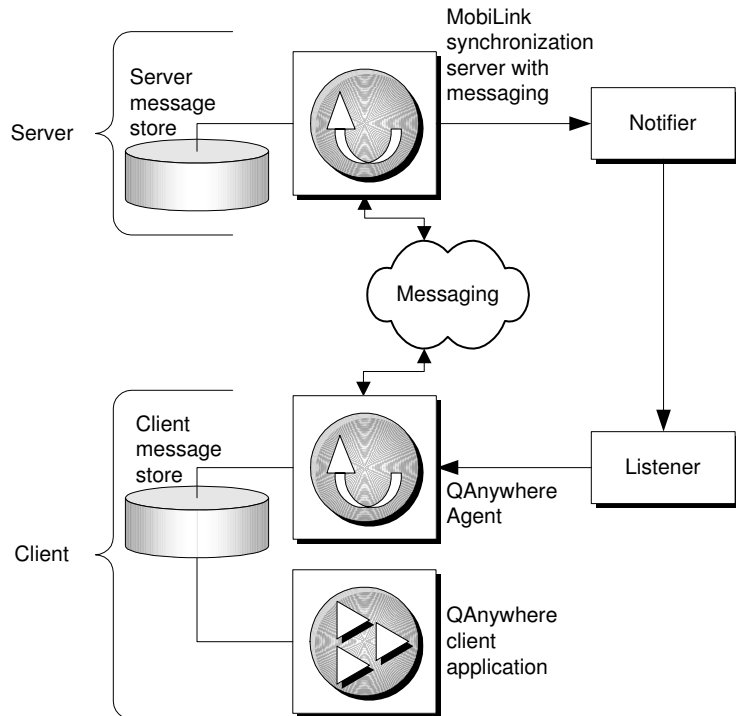
A push notification is a special message delivered from the server to a QAnywhere client. The push notification occurs when a message arrives at

the server message store, and it prompts the client to initiate a message transmission that picks up messages that are ready for the client at the server.

Push notifications introduce two extra components to the QAnywhere architecture. At the server, a QAnywhere Notifier sends push notifications. At the client, a QAnywhere Listener receives these push notifications and passes them on to the QAnywhere Agent.

If you do not use push notifications, messages are still transmitted from the server message store to the client message store, but the transmission must be initiated at the client, such as by using a scheduled transmission.

The architecture for messaging with push notifications is an extension of that described in “[Simple messaging scenario](#)” on page 5. It looks like this:



The components that are added to the “[Simple messaging scenario](#)” on page 5 in order to enable push notification are as follows:

- ◆ **QAnywhere Notifier** The Notifier is a component of the MobiLink synchronization server that is used to deliver push notifications.

The QAnywhere Notifier is a specially configured instance of the Notifier that sends push notifications when a message is ready for delivery.

- 
- ◆ **QAnywhere Listener** The QAnywhere Listener is a separate process that runs at the client. It receives push notifications and passes them on to the QAnywhere Agent.

See also

☞ For more information, see:

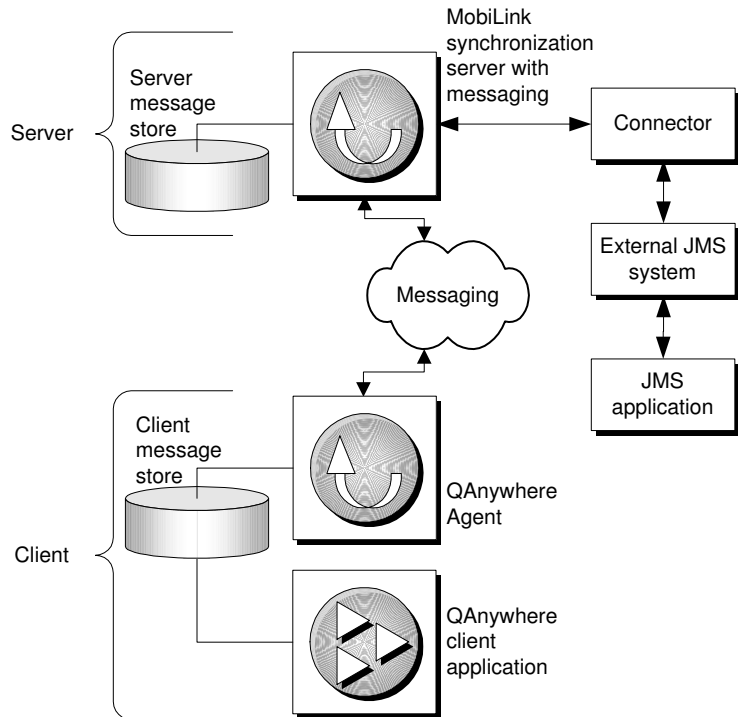
- ◆ [“Using push notifications” on page 40](#)
- ◆ [“Receiving messages asynchronously” on page 70](#)
- ◆ [“Introducing Server-Initiated Synchronization” \[MobiLink Server-Initiated Synchronization User’s Guide, page 1\]](#)

## Scenario for messaging with external messaging systems

In addition to exchanging messages among QAnywhere applications, you can exchange messages with systems that have a JMS interface using a specially configured client known as a connector. JMS is the Java Message Service API for adding messaging capabilities to Java applications.

The external messaging system is set up to act like a special client. It has its own address and configuration.

The architecture for messaging with external messaging systems is an extension of that described in [“Simple messaging scenario” on page 5](#). It looks like this:



The component that is added to “[Simple messaging scenario](#)” on page 5 in order to enable messaging with an external messaging system is as follows:

- ◆ **QAnywhere JMS Connector** The JMS Connector provides an interface between QAnywhere and the external messaging system.

The JMS Connector is a special QAnywhere client that moves messages between QAnywhere and the external JMS system.

See also

For more information, see:

- ◆ “[Using JMS Connectors](#)” on page 42
- ◆ “[Lesson 6: Start a JMS connector](#)” on page 27

---

## Quick start

Following are the steps to set up and run a QAnywhere messaging system.

### ❖ To set up and run QAnywhere messaging

1. Set up a server message store or use an existing MobiLink consolidated database.

☞ See [“Setting up the server message store” on page 32](#).

2. Start the MobiLink synchronization server with the -m option and a connection to the server message store.

☞ See [“Starting the MobiLink synchronization server for QAnywhere messaging” on page 33](#).

3. Set up client message stores. These are Adaptive Server Anywhere databases that are used to temporarily store messages.

☞ See [“Setting up the client message store” on page 35](#).

4. If you want to integrate with an external JMS messaging system, set up JMS messaging for QAnywhere.

☞ See [“Using JMS Connectors” on page 42](#).

5. For each client, write a messaging application.

☞ See [“Writing QAnywhere Client Applications” on page 55](#).

6. For each client, start the QAnywhere Agent with a connection to the local client message store.

☞ See [“Running the QAnywhere Agent” on page 37](#).

Other resources for getting started

- ◆ [“Tutorial: A Sample QAnywhere Application” on page 11](#)
- ◆ Sample applications are installed to *Samples\QAnywhere* in your SQL Anywhere Studio installation directory.



---

## CHAPTER 2

# Tutorial: A Sample QAnywhere Application

### About this chapter

This tutorial explores the capabilities of QAnywhere through a sample client application named TestMessage. QAnywhere applications can run on many devices, such as PDAs, extending application-to-application messaging to these devices. However, for demonstration purposes, this tutorial runs the client on a Windows personal computer.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">About the tutorial</a>	12
<a href="#">Lesson 1: Start MobiLink with messaging</a>	13
<a href="#">Lesson 2: Create a client message store</a>	16
<a href="#">Lesson 3: Run the TestMessage application</a>	18
<a href="#">Lesson 4: Send a message</a>	20
<a href="#">Lesson 5: Explore the TestMessage client source code</a>	22
<a href="#">Lesson 6: Start a JMS connector</a>	27
<a href="#">Tutorial cleanup</a>	29

---

## About the tutorial

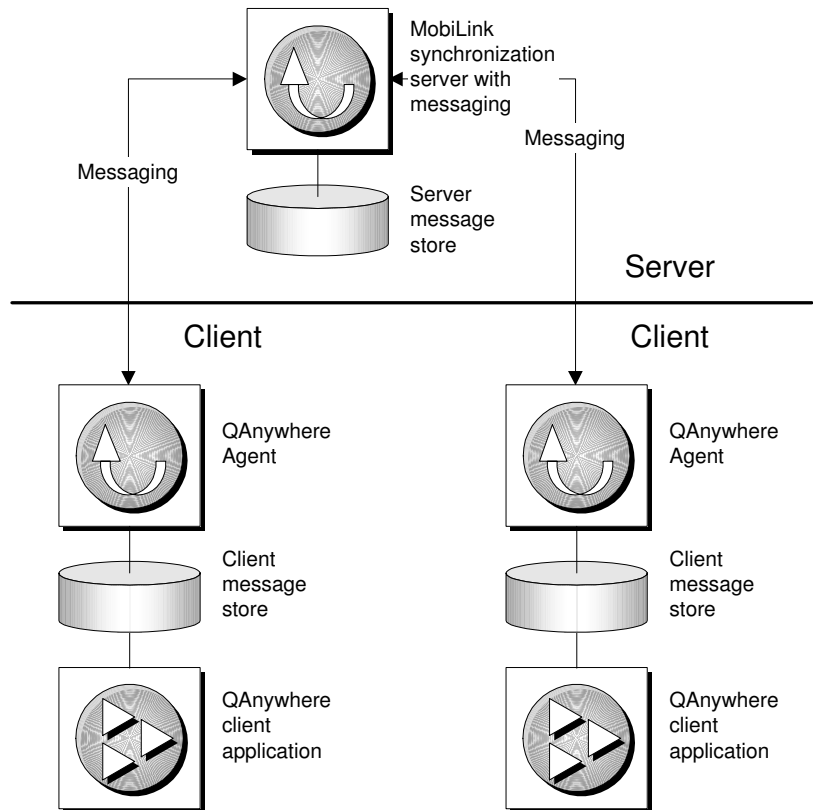
TestMessage is a sample QAnywhere client application. This application demonstrates how you can use QAnywhere to create your own messaging client applications. TestMessage provides a single client-to-client interface to send, receive, and display messages. Being human-readable, text messages provide a useful demonstration of QAnywhere messaging, but QAnywhere provides much more than text messaging. It is a general purpose application-to-application messaging system that provides message-based communication among many clients.

The tutorial is written for a Windows NT/2000/XP system. While these platforms are convenient for demonstration purposes, you can also use QAnywhere to write applications that run on Windows CE devices.

## Lesson 1: Start MobiLink with messaging

### Background

QAnywhere uses MobiLink synchronization to send and receive messages. All messages from one client to another are delivered through a central MobiLink synchronization server. The architecture of a typical system, with only two QAnywhere clients, is shown in the following diagram.



The server message store is a database configured for use as a MobiLink consolidated database. The TestMessage sample uses an Adaptive Server Anywhere consolidated database as its server message store.

The only tables needed in the server message store are MobiLink system tables that are automatically added to any Adaptive Server Anywhere database when it is created. Any supported database that is set up as a MobiLink consolidated database also has these system tables.

The system tables are maintained by MobiLink. Using a relational database as a message store provides a secure, high performance store and means that you can easily integrate messaging into an existing data management and

---

synchronization system.

QAnywhere messaging is usually carried out over separate machines, but in this tutorial all components are running on a single machine. It is important to keep track of which activities are client activities and which are server activities.

In this lesson, you are carrying out actions at the server.

#### Activity

The MobiLink synchronization server can be started with messaging by supplying the `-m` option, as well as specifying a connection string to the server message store. The `TestMessage` sample uses a QAnywhere Adaptive Server Anywhere sample database for the server message store. For the `TestMessage` sample, you can start the MobiLink synchronization server for messaging using the command line options, or using a sample shortcut in your SQL Anywhere Studio install.




#### ❖ Start the messaging server


1. From the Windows Start menu, choose Programs ► SQL Anywhere 9 ► MobiLink ► MobiLink with Messaging Sample.

Alternatively, from a command prompt, navigate to `Samples\QAnywhere\server` subdirectory of your SQL Anywhere Studio installation and type the following command:

```
dbmlsrv9 -m qanysevr.props -c "dsn=QAnywhere 9.0 Sample" -  
vcrs -zu+
```

This example uses the following `dbmlsrv9` options:


Option	Description
-m	The <code>-m</code> option enables messaging. It also specifies the file <code>qanysevr.props</code> , which contains some sample messaging property settings.  See “-m option” [ <i>MobiLink Administration Guide</i> , page 201].
-c	The <code>-c</code> option specifies the connection string to the server message store, in this case using the QAnywhere 9.0 Sample ODBC data source.  See “-c option” [ <i>MobiLink Administration Guide</i> , page 196].
-vcrs	The <code>-vcrs</code> option provides verbose logging of server activities, which is useful during development.  See “-v option” [ <i>MobiLink Administration Guide</i> , page 211].

Option	Description
-zu+	The -zu+ option automatically adds user names to the system; this is convenient for a tutorial or development but is not normally used in a production environment.  See “-zu option” [ <i>MobiLink Administration Guide</i> , page 222].

2. Move the MobiLink synchronization server window to the left side of your screen, which represents the server machine in this tutorial.

Once the MobiLink synchronization server is started and its console window is displaying the message “Ready to handle requests”, you are ready to move on to the next lesson.

#### Further reading

 For more information, see:

- ◆ “Starting the MobiLink synchronization server for QAnywhere messaging” on page 33
- ◆ “-m option” [*MobiLink Administration Guide*, page 201]
- ◆ “Quick start” on page 10
- ◆ “Simple messaging scenario” on page 5

---

## Lesson 2: Create a client message store

### Background

The QAnywhere Agent is a component that runs on each client device. It manages the transmission of messages by moving messages between server message stores and client message stores. The client message store is an Adaptive Server Anywhere database.

The QAnywhere Agent is designed to be running at all times when the device is turned on. QAnywhere applications, in contrast, may be launched and shut down at any time.

In this lesson, you are carrying out activities at a client. Typically, clients run on separate machines from the server, but in this lesson you can create them on the same machine.

In this lesson you will create a client message store.

### Activity

#### ❖ Create a client message store

1. Create an Adaptive Server Anywhere database.

There are several ways to create a database, but in this lesson you will use the `dbinit` command line utility. Navigate to a directory (for example, `c:\sample\qanywhere`) and type:

```
dbinit -I clientstore.db
```

The `dbinit -I` option causes the database to be smaller, which is better for many remote devices.

2. Initialize the database as a client message store.

Type:

```
qaagent -si -c "DBF=clientstore.db" -id MyclientID
```

This example uses the following options:

Option	Description
-si	The -si option initializes an Adaptive Server Anywhere database for use as a client message store. ☞ See “-si option” on page 91.
-c	The -c option specifies the connection string to the client message store. The connection string that is supplied specifies the database file name as <i>clientstore.db</i> . ☞ See “-c option” on page 82.
-id	The -id option specifies an ID for the client message store. Every time you connect to this client message store, you must specify this ID. ☞ See “-id option” on page 84.

The QAnywhere Agent automatically shuts down after initializing a client message store.

#### Further reading

- ☞ For more information about creating a client message store, see:
  - ◆ “Initialization utility options” [*ASA Database Administration Guide*, page 532]
  - ◆ “Setting up the client message store” on page 35

---

## Lesson 3: Run the TestMessage application

### Background

TestMessage is a simple application that uses QAnywhere to send and receive text messages. Text messaging is used in this tutorial because it provides a simple and accessible demonstration of messaging. QAnywhere is, however, not just a text messaging system; it provides general purpose application-to-application messaging.

In this lesson, you are carrying out activities at a client. Typically, clients run on separate machines from the server.

In this Lesson, you start the client message store that is part of the TestMessage sample. In Lesson 4, you will use this message store to send a message to the client message store that you created in Lesson 2.

### Activity

#### ❖ Start the QAnywhere Agent with the TestMessage client message store

1. From the Windows Start menu, choose Programs ► SQL Anywhere Studio 9 ► QAnywhere ► QAnywhere Agent.

This starts the TestMessage sample client message store.

2. The QAnywhere Agent window displays the client message store ID, which by default is your machine name. Make a note of the ID.
3. Move the QAnywhere Agent window to the right side of your screen, which represents the client machine in this tutorial.

#### ❖ Start TestMessage

1. From the Windows Start menu, choose Programs ► SQL Anywhere 9 ► QAnywhere ► TestMessage Sample Application.

The TestMessage window is displayed. The application is connected to the TestMessage client message store that you started in the above procedure.

2. Move the TestMessage window to the right side of your screen, together with the QAnywhere Agent. Both these components belong on the client.
3. Set a preferred name and check the message queue.

From the TestMessage Tools menu, choose Options. Enter a preferred name, which is the name displayed when messages are sent. This name may include spaces.

You will see that the queue name **testmessage** is specified. Do not change this name.



Discussion

You will see messages scrolling by in the MobiLink synchronization server window. This shows that the QAnywhere Agent is periodically transmitting messages between the server message store and client message store.

In a production environment there is generally no need for the frequent transmission activity you see in this tutorial. You can configure the way that the QAnywhere Agent monitors messages by setting a message transmission policy on the command line. The default policy setting is **scheduled**, which instructs the QAnywhere Agent to transmit periodically. If you don't specify an interval, the default is every 10 seconds. Other settings include **automatic**, which sets the QAnywhere Agent to send messages as soon as they are entered in the client message store, **ondemand**, which causes a message to be sent only when instructed to by an application, and a **custom** mode in which you provide a set of rules in a rules file to specify more complicated transmission behavior.

QAnywhere messages are delivered to a QAnywhere address, which consists of a client message store ID and a queue name. The default ID is the machine name on which the QAnywhere Agent is running. Each machine requires only one QAnywhere Agent, even if there are several messaging applications running on the machine. Each application can listen to multiple queues, but each queue should be specific to a single application.

Further reading

- ◆ [“Running the QAnywhere Agent” on page 37](#)
- ◆ [“Determining when message transmission should occur on the client” on page 37](#)
- ◆ [“QAnywhere Agent syntax” on page 80](#)
- ◆ [“QAnywhere Transmission Rules” on page 101](#)
- ◆ [“Writing QAnywhere Client Applications” on page 55](#)
- ◆ The QAnywhere samples, which are installed to *Samples\QAnywhere* in your SQL Anywhere Studio directory

---

## Lesson 4: Send a message

### Background

The TestMessage sample includes a client message store, which you started in Lesson 3. In addition, you created a client message store in Lesson 2. In this lesson you will send a message from the TestMessage sample store to the client message store that you created in Lesson 2.

### Activity

#### ❖ Send a message from TestMessage

1. From the TestMessage Message menu, choose New. The New Message window appears.
2. In the To field, enter MyclientID. This is the ID that you specified for the client message store that you created in Lesson 2.  
QAnywhere appends the queue name specified in the Options dialog to the ID to create a message address. If no queue name is specified in the Options dialog, TestMessage appends the queue name testmessage to the address.
3. Fill out the Subject and Message fields with sample text, and click Send.  
When testing messaging, it is often useful to use the current time as a subject line to make it easy to track individual messages.
4. Shut down TestMessage and the QAnywhere Agent. You should wait at least 10 seconds before doing this.  
In the TestMessage window, click File ► Exit.  
On the QAnywhere Agent window, click Shutdown.
5. Start the QAnywhere Agent with a connection to the client message store that you created in Lesson 2.  
To do this, navigate to the directory where you created the client message store in Lesson 2 and type:

```
qaagent -c "DBF=clientstore.db;eng=qanywhere" -id MyclientID
```

This example uses the following options:

Option	Description
-c	The connection string in this example connects to the client message store that you created in Lesson 2, called <i>clientstore.-db</i> . It specifies <code>eng=qanywhere</code> because the TestMessage sample will attempt to connect to a message store with the database server name <code>qanywhere</code> .
-id	You need to specify <code>MyclientID</code> as the ID because this is the ID you gave this client message store in Lesson 2.

#### 6. Start TestMessage.

From the Windows Start menu, choose Programs ► SQL Anywhere 9 ► QAnywhere ► TestMessage Sample Application.

Your message appears in the TestMessage window. (If your message does not appear, you probably shut down the application too quickly in Step 4.)

#### 7. Read the message.

Select the message to display its contents in the bottom pane of the window.

The next time you start TestMessage, the message will not appear, as TestMessage is configured to delete messages once you have read them. You can change this default behavior by specifying delete rules.

#### Discussion

Like other QAnywhere applications, TestMessage uses the QAnywhere API to manage messages. The QAnywhere API is supplied as both a C++ API and as a Microsoft .NET API that can be used by Visual Basic .NET, C#, and C++ applications developed using Microsoft Visual Studio .NET.

#### Further reading

☞ For more information, see:

- ◆ [“Understanding QAnywhere message addresses” on page 59](#)
- ◆ [“Sending QAnywhere messages” on page 67](#)
- ◆ [“Delete rules” on page 118](#)

---

## Lesson 5: Explore the TestMessage client source code

### Background

This section of the tutorial takes you on a brief tour of the source code behind the TestMessage client application.

A good deal of the code implements the Windows interface, through which you can send, receive, and view the messages. This portion of the tutorial, however, focuses on the portions of the code given to QAnywhere.

You can find the TestMessage source code in the *Samples\QAnywhere* subdirectory of your SQL Anywhere Studio installation.

Several versions of the TestMessage source code are provided. The following versions are provided for Windows 2000 and Windows XP:

- ◆ A C++ version built using the Microsoft Foundation Classes is provided as *Samples\QAnywhere\Desktop\MFC\TestMessage\TestMessage.sln*.
- ◆ A Visual Basic .NET version built on the .NET Framework is provided as *Samples\QAnywhere\Desktop\.NET\VB\TestMessage\TestMessage.sln*.
- ◆ A C# version built on the .NET Framework is provided as *Samples\QAnywhere\Desktop\.NET\CS\TestMessage\TestMessage.sln*.
- ◆ A C++ version built on the .NET Framework is provided as *Samples\QAnywhere\Desktop\.NET\CPP\TestMessage\TestMessage.sln*.

The following version is provided for Pocket PC:

- ◆ A C# version built on the .NET Compact Framework is provided as *Samples\QAnywhere\PocketPC\.NET\CS\TestMessage\TestMessage.sln*.

### Required software

Visual Studio .NET 2003 is required to open the solution files and build the .NET Framework projects and the .NET Compact Framework project.

### Exploring the C# or Visual Basic .NET source

This section takes you through the C# source code. The two versions are structured in a very similar manner.

Rather than look at each line in the application, this lesson picks out certain lines that are particularly useful for understanding QAnywhere applications. It uses the C# version to illustrate particular lines.

1. Open the version of the TestMessage project that you are interested in.

Double-click the solution file to open the project in Visual Studio .NET. For example,

*Samples\QAnywhere\Desktop\.NET\CS\TestMessage\TestMessage.sln* is a solution file. There are several solution files for different environments.

2. Ensure the Solution Explorer is displayed.

You can open the Solution Explorer from the View menu.

3. Inspect the Source Files folder.

There are two files of particular importance. The *MessageList* file (*MessageList.cs*) receives messages and lets you view them. The *NewMessage* file (*NewMessage.cs*) allows you to construct and send messages.

4. From the Solution Explorer, open the *MessageList* file.

5. Inspect the included namespaces.

Every QAnywhere application requires the *iAnywhere.QAnywhere.Client* namespace. The assembly that defines this namespace is supplied as the DLL *iAnywhere.QAnywhere.Client.dll*, in the *ce* or *win32* subdirectory of your SQL Anywhere Studio installation. For your own projects, you must add this DLL as a reference. The namespace is included using the following line at the top of each file:

```
using iAnywhere.QAnywhere.Client;
```

6. Inspect the *MessageList\_Load* method.

This method performs initialization tasks that are common to QAnywhere applications:

- ◆ Create a *QAManager* object.

```
_qaManager =  
QAManagerFactory.Instance.CreateQAManager( null );
```

QAnywhere provides a *QAManagerFactory* object to create *QAManager* objects. The *QAManager* object handles QAnywhere messaging operations: in particular, receiving messages (getting messages from a queue) and sending messages (putting messages on a queue).

QAnywhere provides two types of manager: *QAManager* and *QATransactionalManager*. The difference is that when using the transactional manager all send and receive operations occur within a transaction, so that either all messages are sent (or received) or none are.

- ◆ Write a method to handle messages.

The *onMessage* function that handles regular non-system messages calls the *addMessage* function. The message it receives is encoded as a *QAMessage* object. The *QAMessage* class together with its children (*QATextMessage* and *QABinaryMessage*) provide properties and

---

methods that hold all the information QAnywhere applications need about a message.

```
private void onMessage(QAMessage msg)
{
    if( addMessage( msg ) ) {
        String info_msg = _resources.GetString(
            "MessageReceived" );
        MessageBox.Show( this, info_msg, "Test
            Message",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information );
    }
}
```

◆ Declare a MessageListener.

```
_receiveListener = new
    QAManager.MessageListener( onMessage );
```

The OnMessage method is called whenever a message is received by the QAnywhere Agent and placed in the queue that the application listens to.

**Message listeners and notification listeners**

Message listeners are different from the Listener component described in [“Scenario for messaging with push notifications” on page 6](#). The Listener component receives notifications, while message listener objects retrieve messages from the queue.

7. Inspect the startReceiver method in the same file.

This step assigns message listeners to queues. When you set a message listener for the queue, the QAnywhere Manager will pass messages that arrive on that queue to that listener. Only one listener can be set for a given queue. Setting with a null listener clears out any listener for that queue.

Using a MessageListener is an **asynchronous** way of receiving messages. You can also receive messages **synchronously**; that is, the application explicitly goes and looks for messages on the queue, perhaps in response to a user action such as clicking a Refresh button, rather than being notified when messages appear.

This method completes the initialization tasks:

◆ Open and start the QAManager object.

```
_qaManager.Open(
    AcknowledgementMode.EXPLICIT_ACKNOWLEDGEMENT );
_qaManager.Start();
```

The AcknowledgementMode enumeration constants determine how the receipt of messages is acknowledged to the sender. The

EXPLICIT\_ACKNOWLEDGEMENT constant indicates that messages are not acknowledged until a call to one of the QAManager acknowledge methods is made.

- ◆ Load any messages that are waiting in the queue.

```
_mainWindow.LoadMessages();
```

- ◆ Assign a listener to a queue for future messages.

The listener was declared in the MessageList\_Load method.

```
_qaManager.SetMessageListener(
    Options.GetReceiveQueueName(),
    _receiveListener );
```

The Options.GetReceiveQueueName() function returns the string **testmessage**, which is the TestMessage queue as set in the TestMessage Options dialog.

8. Inspect the addMessage function in the same file.

This method is called whenever the application receives a message. It gets properties of the message such as its ReplyToAddress, PreferredName, and the time it was sent (Timestamp), and displays the information in the TestMessage user interface. Here are the lines that cast the incoming message into a QATestMessage object and get the ReplyToAddress of the message:

```
text_msg = ( QATestMessage )msg;
from = text_msg.ReplyToAddress;
```

This completes a brief look at some of the major tasks carried out in the *MessageList* file.

9. From the Solution Explorer, open the *NewMessage* file.

10. Inspect the sendMessage function.

This function takes the information entered in the New Message dialog and constructs a QATestMessage object. The QAManager then puts the message in the queue to be sent.

Here are the lines that create a QATestMessage object and set its ReplyToAddress property:

```
qa_manager = MessageList.GetQAManager();
msg = qa_manager.CreateTextMessage();
msg.ReplyToAddress = Options.GetReceiveQueueName();
```

Here are the lines that put the message in the queue to be sent. The variable `to` is the destination address, supplied as an argument to the function.

---

```
to = BuildQueue( to );  
qa_manager.PutMessage( to, msg );
```

#### Further reading

☞ For more information, see:

- ◆ [“QAnywhere C++ API Reference” on page 121](#)
- ◆ [“iAnywhere.QAnywhere.Client namespace” on page 173](#)
- ◆ [“Writing QAnywhere Client Applications” on page 55](#)
- ◆ The TestMessage sample, which is installed to *Samples\QAnywhere* in your SQL Anywhere Studio directory



## Lesson 6: Start a JMS connector

A JMS connector provides connectivity between a JMS message system and QAnywhere.

### Required software

For this lesson, you need access to a JMS provider and basic knowledge of how to configure it. In addition, you need JDK version 1.3.1 or later and any JAR files required by a JMS client of the JMS provider.

### Activity

#### ❖ Prepare your JMS provider

1. Start your JMS server.

☞ See the documentation for your JMS server.

2. Create two queues within your JMS server: `qa_testmessage` and `qa_receive`.

☞ See the documentation for your JMS server. You may need to restart your JMS server after creating the queues.

#### ❖ Start QAnywhere client and server components

1. Create a QAnywhere connector properties file.

There are sample connector properties files in the `Samples\QAnywhere\connectors` subdirectory of your SQL Anywhere Studio installation.

2. Specify your QAnywhere connector properties file in the file `Samples\QAnywhere\server\qanyserv.props`.

This sample file contains commented lines that specify the sample connector properties files. You may just need to uncomment one of these lines.

3. Start the MobiLink synchronization server for messaging, as described in [“Lesson 1: Start MobiLink with messaging”](#) on page 13.
4. Start QAnywhere Agent as described in [“Lesson 2: Create a client message store”](#) on page 16.
5. Start TestMessage as described in [“Lesson 3: Run the TestMessage application”](#) on page 18.

---

## ❖ Start the TestMessage client

1. At a command prompt, navigate to *Samples\QAnywhere\connectors\JMS\TestMessage* and type the following:

```
java -cp .;JMS-client-jar-files
      anywhere.message.samples.TestMessage
```

where *JMS-client-jar-files* is a semi-colon delimited list of jar files that the JMS server requires. See your JMS server documentation for details. For Sybase EAServer, this command would be:

```
java -cp .; path\easclient.jar;path\easj2ee.jar
      anywhere.message.samples.TestMessage
```

where *path* is the location of the jar files.

2. Move the JMS TestMessage window to the right side of your screen under the existing TestMessage window.
3. From the JMS TestMessage Message menu, choose New.  
The New Message window appears.
4. In the To field, enter the client message store ID that you noted in Lesson 2.
5. Fill out the Subject and Message fields with sample text, and click Send.  
Within a short time a message box appears, indicating that a message has been received by the previously running instance of TestMessage.
6. Switch to the other instance of TestMessage to receive the message.

### Further reading

☞ For more information, see:

- ◆ [“Using JMS Connectors” on page 42](#)
- ◆ [“Scenario for messaging with external messaging systems” on page 8](#)
- ◆ The JMS sample properties files and other JMS samples that are installed to *Samples\QAnywhere\connectors* in your SQL Anywhere Studio directory

## Tutorial cleanup

Shut down TestMessage, the QAnywhere Agent, and the MobiLink synchronization server.



---

## CHAPTER 3

# Setting Up QAnywhere Messaging

### About this chapter

This chapter describes how to set up and run QAnywhere messaging.

QAnywhere uses MobiLink synchronization to transport messages. This chapter describes how to set up and run the MobiLink synchronization server with messaging.

### Contents

<b>Topic:</b>	<b>page</b>
<a href="#">Setting up server-side components</a>	32
<a href="#">Setting up client-side components</a>	35
<a href="#">Using push notifications</a>	40
<a href="#">Using JMS Connectors</a>	42
<a href="#">Using QAnywhere messaging and MobiLink data synchronization together</a>	51
<a href="#">Setting up a failover mechanism</a>	53

---

# Setting up server-side components

## ❖ Overview of setting up QAnywhere server-side components

1. Set up a server message store and start it. This can be any MobiLink consolidated database.  
☞ See [“Setting up the server message store” on page 32](#).
2. Start dbmlsrv9 with the -m option and a connection to the server message store.  
☞ See [“Starting the MobiLink synchronization server for QAnywhere messaging” on page 33](#).
3. Add client message store IDs to the server message store.  
☞ See [“Adding client message store IDs” on page 34](#).

## Setting up the server message store

The server message store is a relational database on the server that temporarily stores messages until they are transmitted to a client message store or JMS system. Messages are exchanged between clients via the server message store.

A server message store is a MobiLink consolidated database, and so can be any RDBMS that MobiLink supports (Adaptive Server Anywhere, Adaptive Server Enterprise, Microsoft SQL Server, Oracle, or DB2). You can create a new database for this purpose, or use an existing database. The database does not need to be dedicated to acting as a server message store and so can also be used for other purposes.

Adaptive Server Anywhere databases are automatically configured so that they can be used as consolidated databases.

☞ For information about creating Adaptive Server Anywhere databases, see [“The Initialization utility” \[ASA Database Administration Guide, page 530\]](#).

If you are using an Adaptive Server Anywhere database that was created before version 9.0.2, it must be upgraded.

☞ To upgrade your database, see [“Upgrading Software and Databases” \[What’s New in SQL Anywhere Studio, page 227\]](#).

For databases other than Adaptive Server Anywhere, you need to run a setup script that enables the database to be used as a consolidated database (if you have not already done so).

- ☞ To set up a server message store for databases other than Adaptive Server Anywhere, see [“Setting up a consolidated database” \[MobiLink Administration Guide, page 33\]](#).
- Notes
- ◆ You can integrate messaging with MobiLink synchronization applications by using your MobiLink consolidated database as the server message store.
    - ☞ See [“Using QAnywhere messaging and MobiLink data synchronization together” on page 51](#).
- Example
- To create an Adaptive Server Anywhere database called `qanysrv.db`, type the following at a command prompt:
- ```
dbinit qanysrv.db
```
- This database is ready to use as a server message store.

## Starting the MobiLink synchronization server for QAnywhere messaging

QAnywhere uses MobiLink synchronization to transport messages. To use QAnywhere messaging, you must start the MobiLink synchronization server (dbmlsrv9) with the following options:

- ◆ **-c connection-string** To connect to the server message store.
  - ☞ See [“-c option” \[MobiLink Administration Guide, page 196\]](#).
- ◆ **-m** To enable QAnywhere messaging. The -m option also allows you to optionally specify configuration information.
  - ☞ See [“-m option” \[MobiLink Administration Guide, page 201\]](#).
- ◆ **-zu+** Optionally, you may want to specify -zu+ while in development mode. When you create client message stores, the client message store ID must be added to the consolidated database, and -zu+ allows this to happen automatically.
  - ☞ See [“-zu option” \[MobiLink Administration Guide, page 222\]](#).

☞ You can also use other MobiLink synchronization server options to customize your operations. For more information, see [“MobiLink synchronization server” \[MobiLink Administration Guide, page 190\]](#).

- Notes
- ◆ If you are integrating with a JMS messaging system, there are other options you must specify when you start the MobiLink synchronization server.
    - ☞ See [“Starting the MobiLink server for JMS integration” on page 43](#).

---

## Example

To start QAnywhere messaging when you are using a server message store called *qanysrv.db*, type the following at a command prompt:

```
dbmlsrv9 -m -c "dbf=qanysrv.db"
```

## Adding client message store IDs

Each client message store has a unique ID that identifies it. The client message store ID is a MobiLink user name. You may need to add this MobiLink user name to the server message store. There are several methods for doing this:

- ◆ Use the `dbmluser` utility.
  - ☞ For more information, see [“MobiLink user authentication utility”](#) [*MobiLink Administration Guide*, page 492].
- ◆ Use Sybase Central.
- ◆ Specify the `-zu+` command line option with `dbmlsrv9`. In this case, any existing MobiLink users that have not been added to the consolidated database are added when they first synchronize. This is useful during development, but is not recommended for production environments.
  - ☞ For more information, see [“-zu option”](#) [*MobiLink Administration Guide*, page 222].
  - ☞ For more information about MobiLink user names, see [“About MobiLink users”](#) [*MobiLink Clients*, page 10].
  - ☞ For more information about client message store IDs, see [“-id option”](#) on page 84.



## Setting up client-side components

### ❖ Overview of setting up client-side components

1. Create an Adaptive Server Anywhere database and initialize it as a client message store.
  - ☞ See [“Setting up the client message store”](#) on page 35.
2. Ensure that the ID of the client message store is registered on the server message store.
  - ☞ See [“Adding client message store IDs”](#) on page 34.
3. Write client applications.
  - ☞ See [“Writing QAnywhere Client Applications”](#) on page 55.
4. Start the QAnywhere Agent.
  - ☞ See [“Running the QAnywhere Agent”](#) on page 37.

### Setting up the client message store

The client message store is an Adaptive Server Anywhere database on the remote device. The application connects to this message store using the QAnywhere client API.

The database that you use for the client message store must not be used by any other non-messaging applications. However, you can use a MobiLink remote database in conjunction with a client message store to integrate messaging with data synchronization.

Using a relational database as a message store provides a secure and high-performance store, and also enables easy integration of messaging with data synchronization.

☞ For more information, see [“Using QAnywhere messaging and MobiLink data synchronization together”](#) on page 51 and [“Creating a secure client message store”](#) on page 96.

### ❖ To create a client message store

1. Create an Adaptive Server Anywhere database.
  - ☞ See [“Creating a database”](#) [*ASA SQL User’s Guide*, page 31].
2. Initialize each client message store by running the QAnywhere Agent (qaagent) with the following options:

---

◆ **-c option** to specify a connection string to the database you just created.

☞ See “[-c option](#)” on page 82.

◆ **-si option** to initialize the database.

☞ See “[-si option](#)” on page 91.

◆ **-id option** optionally, if you want to pre-assign a client message store ID.

☞ See “[Creating client message store IDs](#)” on page 36 and “[-id option](#)” on page 84.

3. Add client message store IDs to the server message store. This can be done automatically using the `dbmlsrv9 -zu+` option, or can be done in other ways.

☞ See “[Adding client message store IDs](#)” on page 34.

4. Change the default passwords and take other steps to ensure that the client message store is secure.

☞ See “[Creating a secure client message store](#)” on page 96.

You can also upgrade a client message store that was created in a previous version of QAnywhere.

☞ See “[-su option](#)” on page 92.

## Creating client message store IDs

Client message store IDs can be set in various ways:

- ◆ You can specify the ID with the `qaagent -id` option when you use the `qaagent -si` option to initialize the client message store.
- ◆ You can specify the ID with the `-id` option the first time you run `qaagent` after you initialize the client message store.
- ◆ If you do not specify an ID in either of the previous ways, then the first time you run `qaagent` after you run `qaagent` with `-si`, the device name is assigned as the client message store ID.

☞ For more information, see “[QAnywhere Agent](#)” on page 79.

## Example of creating a client message store

The following command creates an Adaptive Server Anywhere database called `qanyclient.db`. (The `dbinit -i` option is not required, but is good practice as it reduces the size of the database.)

```
dbinit -i qanyclient.db
```

The following command connects to `qanyclient.db` and initializes it as a QAnywhere client database:

```
qaagent -si -c "DBF=qanyclient.db"
```

☞ For more information about dbinit, see “Creating a database using the dbinit command-line utility” [ASA Database Administration Guide, page 531].

## Running the QAnywhere Agent

The QAnywhere Agent (qaagent) is a separate process running on the client device that monitors the client message store and determines when message transmission should occur.

The QAnywhere Agent transmits messages between the server message store and the client message store. You can run only one instance of the QAnywhere Agent on a device at a time.

At a minimum, you need to start qaagent with the following options:

- ◆ **-c “connection-string”** to connect to the client message store.
- ◆ **-id client-message-store-id** to identify the client message store. The first time you run qaagent after you have initialized a client message store, you can optionally use this option to name the message store; if you do not, the device name is used by default. After that, you must use the -id option every time you start qaagent to specify the client message store ID.
- ◆ **-x protocol[ (protocol-options) ]** to connect to the MobiLink synchronization server (necessary unless the MobiLink synchronization server is running on the same device as the QAnywhere agent).

☞ For details of these options as well as a complete list of all qaagent options, see “QAnywhere Agent syntax” on page 80.

## Determining when message transmission should occur on the client

On the client side, you determine when message transmission should occur by specifying policies. A policy tells the QAnywhere Agent when a message should be moved from the client message store to the server message store. If you do not specify a policy, transmission occurs every 10 seconds by default. There are three pre-defined policies: scheduled, automatic and on-demand. These policies are specified using the qaagent -policy option.

You can also define a custom policy. A custom policy is a set of rules that determine when a message transmission is to occur. In addition, a custom policy can specify which messages should be transmitted. A custom policy is used when a file containing the transmission rules is specified using the -policy option.

### Scheduled policy

The scheduled policy instructs the Agent to perform a transmission at a specified time interval. To invoke a schedule, use the **scheduled** keyword

---

when you start the QAnywhere Agent:

**qaagent –policy scheduled** [ *interval* ] ...

where *interval* is in seconds. The default is 10 seconds.

When a schedule is specified, every *n* seconds the Agent performs a message transmission if any of the following conditions are met:

- ◆ New messages were placed in the client message store since the previous time interval elapsed.
- ◆ A message status change occurred since the previous time interval elapsed. This typically occurs when a message is acknowledged by the application.
- ◆ A push notification was received since the previous time interval elapsed.
- ◆ A network status change notification was received since the previous time interval elapsed.
- ◆ Push notifications are disabled.

You can call the `TriggerSendReceive()` method to override the time interval. It forces a message transmission to occur before the time interval elapses.

#### Automatic policy

The automatic policy attempts to keep the client and server message stores as up-to-date as possible. To set up automatic message transmission, use the **automatic** keyword when you start the QAnywhere Agent:

**qaagent –policy automatic**

When using the automatic policy, a message transmission will be performed when any of the following occurs:

- ◆ `PutMessage()` is called.
- ◆ A message status change has occurred. This typically occurs when a message is acknowledged by the application.
- ◆ A Push Notification is received.
- ◆ A Network Status Change Notification is received.
- ◆ `TriggerSendReceive()` is called.

#### On-demand policy

The on-demand policy causes a message transmission to occur only when instructed to do so by an application. To start this mode, use the **ondemand** keyword when you start the QAnywhere Agent:

**qaagent –policy ondemand**

An application forces a message transmission to occur by calling the `TriggerSendReceive()` method.

When the agent receives a Push Notification or a Network Status Change Notification, a corresponding message is sent to the “system” queue. This allows an application to detect these events and force a message transmission by calling the `TriggerSendReceive()` method.

#### Custom policy

A custom policy allows you to define when a message transmission is to occur and which messages are sent in the message transmission. The custom policy is defined by a set of rules stored in a file. To define a custom policy, specify a rules file when you start the QAnywhere Agent:

**qaagent -policy rules-file**

Each rule is of the following form:

*schedule = condition*

where *schedule* defines when *condition* is evaluated. For more information, see [“Schedule syntax” on page 105](#).

All messages satisfying *condition* are transmitted. In particular, if *schedule* is automatic, the condition is evaluated when any of the following occurs.

- ◆ `PutMessage()` is called.
- ◆ A message status change has occurred. This typically occurs when a message is acknowledged by the application.
- ◆ A Push Notification is received.
- ◆ A Network Status Change Notification is received.
- ◆ `TriggerSendReceive ( )` is called.

☞ For more information about specifying a custom policy in a rules file, see [“Transmission rules” on page 102](#).

☞ For more information about creating policies, see [“-policy option” on page 88](#).

---

## Using push notifications

A push notification is a special message delivered from the server to a QAnywhere client that prompts the client to initiate a message transmission. Push notifications introduce two extra components to the QAnywhere architecture:

- ◆ At the server, a QAnywhere Notifier sends push notifications.
- ◆ At the client, a QAnywhere Listener receives these push notifications and passes them on to the QAnywhere Agent.

Notification is enabled by default: the `qaagent -push_notification` option is by default set to enabled. Push notifications are delivered either over the UDP protocol or as SMS messages.

If you are using UDP, push notifications will probably work without any configuration, but due to a limitation in the UDP implementation of ActiveSync, they will not work with ActiveSync.

If you are using SMS, you need to start a Listener to use notifications. For more information, see [“Using push notifications with SMS” on page 40](#).

☞ To disable push notifications, see [“-push\\_notifications option” on page 90](#).

See also

☞ For more information about push notifications, see:

- ◆ [“Scenario for messaging with push notifications” on page 6](#)
- ◆ [“Handling push notifications and network status changes” on page 73](#)
- ◆ [“-push\\_notifications option” on page 90](#)
- ◆ [“Using push notifications with SMS” on page 40](#)

## Using push notifications with SMS

You can use QAnywhere notifications when your message transmission is occurring over SMS, but you must start a Listener using the `dblsn` executable.

☞ For more information, see [“Listeners” \[MobiLink Server-Initiated Synchronization User’s Guide, page 28\]](#).

Start the QAnywhere Agent as you normally would:

```
qaagent -id device-id -c "connect-string"
```

On the server side, set SMTP properties that are required to send SMS messages.

☞ For more information, see “SMTP gateway properties” [[MobiLink Server-Initiated Synchronization User’s Guide](#), page 70].

---

## Using JMS Connectors

JMS is the Java Message Service API for adding messaging capabilities to Java applications. In addition to exchanging messages among QAnywhere client applications, you can exchange messages with external messaging systems that support a JMS interface. You do this using a specially configured client known as a connector. In a QAnywhere application, the external messaging system is set up to act like a QAnywhere client. It has its own address and configuration.

☞ For more information about the architecture of this approach, see [“Scenario for messaging with external messaging systems”](#) on page 8.

### ❖ Overview of integrating a QAnywhere application with an external JMS system

1. Create JMS queues using the JMS administration tools for your JMS system. The QAnywhere connector listens on a single JMS queue for JMS messages. You must create this queue if it does not already exist.  
☞ See the documentation of your JMS product for information about how to create queues.
2. Create a JMS connector properties file and set the `ianywhere.connector.address` property to specify the connector address. This is the address used by QAnywhere applications to send messages through the connector.  
☞ See [“Configuring the JMS connector properties file”](#) on page 43.
3. Create a MobiLink messaging properties file and set the `ianywhere.qa.server.connectorPropertiesFiles` property to the name of the JMS connector properties file.  
☞ For more information about specifying a Mobilink messaging properties file, see [“-m option”](#) [*MobiLink Administration Guide*, page 201].
4. Start the MobiLink synchronization server with a connection to the server message store and the options `-m` and `-sl java`. The MobiLink messaging properties file you created in the previous step is referenced by the `-m` option.  
☞ See [“Starting the MobiLink server for JMS integration”](#) on page 43.
5. To send a message from an application in your QAnywhere system to the external messaging system, create a QAnywhere message and send it to **`connector-address\JMS-queue`**.  
☞ See [“Addressing QAnywhere messages meant for JMS”](#) on page 46.



6. To send a message from the external messaging system to an application in your QAnywhere system, create a JMS message, set the `ias_ToAddress` property to the QAnywhere `id\queue` (where `id` is the ID of your client message store and `queue` is your application queue name), put the message in the JMS queue and use the Send operation.
  - ☞ See “Addressing JMS messages meant for QAnywhere” on page 48.

## Starting the MobiLink server for JMS integration

To exchange messages with an external messaging system that supports a JMS interface, you must start the MobiLink synchronization server (`dbmlsrv9`) with the following options:

- ◆ **-c *connection-string*** To connect to the server message store.
  - ☞ See “-c option” [MobiLink Administration Guide, page 196].
- ◆ **-m *message-properties-file*** The *message-properties-file* must define an `ianywhere.qa.server.connectorPropertiesFiles` property that specifies a JMS Connector properties file or files. The message properties file should contain the following statement:

```
ianywhere.qa.server.connectorPropertiesFiles=file1;[ file2;
]...
```

See “-m option” [MobiLink Administration Guide, page 201].

- ◆ **-sl java (-cp “*jarfile.jar*”)** To add the client jar files required to use the external JMS provider.
  - ☞ See “-sl java option” [MobiLink Administration Guide, page 209].

### Example


The following example starts a MobiLink synchronization server using a message properties file called `msg.props` (in the current working directory), a JMS client library called `jmsclient.jar` (also in the current working directory), and the QAnywhere 9.0 Sample database as a message store. The command should be entered all on one line.

```
dbmlsrv9 -sl java(-cp "jmsclient.jar")
-m msg.props
-c "QAnywhere 9.0 Sample" ...
```

## Configuring the JMS connector properties file

The JMS Connector properties file contains entries that specify connection information with the JMS system. It configures a connector to a third party JMS messaging system such as BEA WebLogic or Sybase EAServer.

The following properties are used to configure the JMS Connector.

- 
- ◆ **ianywhere.connector.nativeConnection** The Java class that implements the connector. This Java class is provided by QAnywhere and should always be set to `ianywhere.message.connector.jms.NativeConnectionJMS`.
  - ◆ **ianywhere.connector.id** An identifier that uniquely identifies the connector. This identifier is also used to prefix all logged error, warning, and informational messages appearing in the QAnywhere server console for this connector.
  - ◆ **ianywhere.connector.address** The connector address that a QAnywhere client should use to address the connector.  
 For more information, see [“Addressing QAnywhere messages meant for JMS” on page 46](#).
  - ◆ **ianywhere.connector.outgoing.deadMessageAddress** The address that a message is sent to when it cannot be processed. For example, if a message contains an address that is malformed or unknown, the message is sent to the dead message address.  
The default dead message address is `ianywhere.connector.deadMessage`.
  - ◆ **ianywhere.connector.logLevel** The amount of connector information displayed in the console and log file. Values for the log level are as follows:
    - 1 Log error messages.
    - 2 Log error and warning messages.
    - 3 Log error, warning, and information messages.
    - 4 Debug mode: provide most verbose logging.
  - ◆ **ianywhere.connector.compressionLevel** The default message compression factor of messages received from JMS: an integer between 0 and 9, with 0 indicating no compression and 9 indicating maximum compression.  
You can set the compression level for all connectors using the `ianywhere.qa.compressionLevel` property in the configuration file that you supply with the `dbmlsrv9 -m` option. If you set the compression level in both places, this setting for an individual connector overrides the setting for all connectors. For more information, see [“-m option” \[MobiLink Administration Guide, page 201\]](#).
  - ◆ **xjms.jndi.authName** The authentication name to connect to the external JMS JNDI name service.
  - ◆ **xjms.jndi.factory** The factory name used to access the external JMS JNDI name service.

- ◆ **xjms.jndi.password.e** The authentication password to connect to the external JMS JNDI name service.
- ◆ **xjms.jndi.url** The URL to access the JMS JNDI name service.
- ◆ **xjms.password.e** The authentication password to connect to the external JMS provider.
- ◆ **xjms.queueFactory** The external JMS provider queue factory name.
- ◆ **xjms.queueConnectionAuthName** The user ID to connect to the external JMS queue connection.
- ◆ **xjms.queueConnectionPassword.e** The password to connect to the external JMS queue connection.
- ◆ **xjms.topicFactory** The external JMS provider topic factory name.
- ◆ **xjms.topicConnectionAuthName** The used ID to connect to the external JMS topic connection.
- ◆ **xjms.topicConnectionPassword.e** The password to connect to the external JMS topic connection.
- ◆ **xjms.receiveDestination** The queue name used by the connector to listen for messages from JMS targeted for QAnywhere clients.

#### Example

You can find sample connector property files in the *Samples\QAnywhere\connectors* subdirectory of your SQL Anywhere Studio installation. Here is a sample properties file for Sybase EAServer:

```
ianywhere.connector.nativeConnection=ianywhere.message.connector
    .jms.NativeConnectionJMS
ianywhere.connector.id=easerver
ianywhere.connector.logLevel=4
ianywhere.connector.address=ianywhere.connector.easerver
xjms.jndi.factory=com.sybase.jms.InitialContextFactory
xjms.jndi.url=iiop://<substitute with host name>:9000
xjms.jndi.authName=jagadmin
xjms.jndi.password.e=
xjms.topicFactory=javax.jms.TopicConnectionFactory
xjms.topicConnectionAuthName=
xjms.topicConnectionPassword.e=
xjms.queueFactory=javax.jms.QueueConnectionFactory
xjms.queueConnectionAuthName=
xjms.queueConnectionPassword.e=
xjms.receiveDestination=qanywhere
```

### Configuring multiple connectors

QAnywhere can connect to multiple JMS message systems by defining a JMS connector file for each JMS system. Each connector to a JMS message

---

system must be configured using its own JMS Connector properties file. The only property values that must be unique among the configured connectors are `ianywhere.connector.id` and `ianywhere.connector.address`.

- ◆ The `ianywhere.connector.id` property must be unique. It is used to prefix all connector messages in the QAnywhere server console.
- ◆ The `ianywhere.connector.address` property is the address prefix that QAnywhere clients must specify to address messages meant for the JMS system.

☞ For information about specifying the address of QAnywhere clients, see [“Addressing QAnywhere messages meant for JMS” on page 46](#).

☞ For information about the connector properties file, see [“Configuring the JMS connector properties file” on page 43](#).

## Addressing QAnywhere messages meant for JMS

A QAnywhere client can send a message to a JMS system by setting the address to `connector_address\JMS_destination`. The `connector_address` is the value of the connector property `ianywhere.connector.address`, while `JMS_destination` is the name used to look up the JMS queue or topic using the Java Naming and Directory Interface.

☞ For more information, see [“Understanding QAnywhere message addresses” on page 59](#).

### Example

For example, if the `ianywhere.connector.address` is set to `ianywhere.connector.easerver` and the JMS queue name is `myqueue`, then the code to set the address would be:

```
C#
QAManagerBase mgr;
QAMessage msg;
// Initialize the manager
...
msg = mgr.CreateTextMessage();
// Set the message content
...
mgr.PutMessage(@"ianywhere.connector.easerver\myqueue", msg );

C++
QAManagerBase *mgr;
QATextMessage *msg;
// Initialize the manager
...
msg = mgr.createTextMessage();
// Set the message content
...
mgr->putMessage( "ianywhere.connector.easerver\\myqueue", msg );
```

## Mapping QAnywhere messages on to JMS messages

QAnywhere messages are mapped naturally on to JMS messages.

QAnywhere message content

| QAnywhere       | JMS                    | Remarks                        |
|-----------------|------------------------|--------------------------------|
| QATextMessage   | javax.jms.TextMessage  | message text copied as Unicode |
| QABinaryMessage | javax.jms.BytesMessage | message bytes copied exactly   |

QAnywhere built-in headers

The following table describes the mapping of built-in headers. In C++ and JMS, these are method names; for example, Address is called getAddress or setAddress for QAnywhere, and getJMSDestination or setJMSDestination for JMS. In .NET, these are properties with the exact name given below; for example, Address is Address.

| QAnywhere                                           | JMS                                               | Remarks                                                                                                                                                                                                                           |
|-----------------------------------------------------|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Address                                             | JMS Destination and JMS property<br>ias_ToAddress | Only the JMS part of the address is mapped to the Destination. Under rare circumstances, in the case of a message looping back into QAnywhere, there may be an additional QAnywhere address suffix. This is put in ias_ToAddress. |
| Expiration                                          | JMS Expiration                                    |                                                                                                                                                                                                                                   |
| InReplyToID                                         | JMS CorrelationID                                 |                                                                                                                                                                                                                                   |
| MessageID                                           | N/A                                               | Not mapped.                                                                                                                                                                                                                       |
| Priority                                            | JMS Priority                                      |                                                                                                                                                                                                                                   |
| Redelivered                                         | N/A                                               | Not mapped.                                                                                                                                                                                                                       |
| ReplyToAddress                                      | JMS property<br>ias_ReplyToAddress                | Mapped to JMS property.                                                                                                                                                                                                           |
| Connector's xjms.-receiveDestination property value | JMS ReplyTo                                       | ReplyTo set to Destination used by connector to receive JMS messages.                                                                                                                                                             |

| QAnywhere | JMS | Remarks     |
|-----------|-----|-------------|
| Timestamp | N/A | Not mapped. |

## QAnywhere properties

QAnywhere properties are all mapped naturally to JMS properties, preserving type, with one exception. If the QAnywhere message has a property called JMSType, then this is mapped to the JMS header property JMSType.

## Addressing JMS messages meant for QAnywhere

A JMS client can send a message to a QAnywhere client by setting the JMS message property `ias_ToAddress` to the QAnywhere address, and then sending the message to the JMS Destination corresponding to the connector property `xjms.receiveDestination`.

☞ For more information, see [“Understanding QAnywhere message addresses”](#) on page 59.

### Example

For example, to send a message to the QAnywhere address “qaddr” (where the connector setting of `xjms.receiveDestination` is “`ianywhere.connector.jms_receive`”):

```
import javax.jms.*;
...
try {
    QueueSession session;
    QueueSender sender;
    TextMessage mgr;
    Queue connectorQueue;
    // Initialize the session
    ...
    connectorQueue = session.createQueue(
        "ianywhere.connector.jms_receive" );
    sender = session.createSender( connectorQueue );
    msg = session.createTextMessage();
    msg.setStringProperty( "ias_ToAddress", "qaddr" );
    // Set the message content
    ...
    sender.send( msg );
} catch( JMSEException e ) {
    // Handle the exception
    ...
}
```

## Mapping JMS messages on to QAnywhere messages

QAnywhere messages are mapped naturally on to JMS messages.

### JMS message content

| JMS                          | QAnywhere       | Remarks                        |
|------------------------------|-----------------|--------------------------------|
| javax.jms.TextMessage        | QATextMessage   | Message text copied as Unicode |
| javax.jms.-<br>BytesMessage  | QABinaryMessage | Message bytes copied exactly   |
| javax.jms.-<br>StreamMessage | N/A             | Not supported                  |
| javax.jms.MapMessage         | N/A             | Not supported                  |
| javax.jms.-<br>ObjectMessage | N/A             | Not supported                  |

### JMS built-in headers

The following table describes the mapping of built-in headers. In C++ and JMS, these are method names; for example, Address is called `getAddress` or `setAddress` for QAnywhere, and `getJMSDestination` or `setJMSDestination` for JMS. In .NET, these are properties with the exact name given below; for example, Address is Address.

| JMS               | QAnywhere   | Remarks                                                                                                                          |
|-------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------|
| JMS Destination   | N/A         | The JMS destination must be set to the queue specified in the connector property <code>ianywhere.-connector.jms_receive</code> . |
| JMS Expiration    | Expiration  |                                                                                                                                  |
| JMS CorrelationID | InReplyToID |                                                                                                                                  |
| JMS MessageID     | N/A         | Not mapped.                                                                                                                      |
| JMS Priority      | Priority    |                                                                                                                                  |
| JMS Redelivered   | N/A         | Not mapped.                                                                                                                      |

| <b>JMS</b>                                                              | <b>QAnywhere</b>                   | <b>Remarks</b>                                                                                |
|-------------------------------------------------------------------------|------------------------------------|-----------------------------------------------------------------------------------------------|
| JMS ReplyTo and connector's ianywhere.-connector.address property value | ReplyToAddress                     | The connector address is concatenated with the JMS ReplyTo Destination name delimited by '\.' |
| JMS DeliveryMode                                                        | N/A                                | Not mapped.                                                                                   |
| JMS Type                                                                | QAnywhere message property JMSType |                                                                                               |
| JMS Timestamp                                                           | N/A                                | Not mapped.                                                                                   |

## JMS properties

JMS properties are all mapped naturally to QAnywhere properties, preserving type, with a few exceptions. The QAnywhere Address property is set from the value of the JMS message property `ias_ToAddress`. If the JMS message property `ias_ReplyToAddress` is set, then the QAnywhere `ReplyToAddress` is additionally suffixed with this value delimited by a '\.'



## Using QAnywhere messaging and MobiLink data synchronization together

You can integrate QAnywhere messaging into MobiLink synchronization applications that use Adaptive Server Anywhere clients.

To do this,

- ◆ The QAnywhere client message store database and the database used by the data synchronization application must be different database files. However, for efficiency on small devices, you can run both databases on the same database server.
- ◆ You must use your MobiLink consolidated database as the server message store. This can be any supported MobiLink consolidated database (Adaptive Server Anywhere, Adaptive Server Enterprise, Oracle, DB2, or Microsoft SQL Server).
- ◆ You can create a separate QAnywhere messaging application or integrate messaging into your data synchronization application.

### Example

The following example integrates QAnywhere messaging into a MobiLink synchronization system that has a server named *MyServer* and an Adaptive Server Anywhere client database file called *MyAppData.db*.

Create the client message store database:

```
dbinit -i MyAppData.db
```

Initialize the client message store:

```
qaagent -id MyID -si -c "dbf=MyAppData.db"
```

Start the database server on the remote device with the *MyAppData.db* and *qanywhere.db* database files.

```
dbsrv9 -n MyServer MyAppData.db -n MyAppData qanywhere.db -n qanywhere
```

Start the QAnywhere Agent, using *qanywhere.db* as the message store:

```
qaagent -id MyID -c "eng=MyServer;dbn=qanywhere"
```

When creating a QAManager instance, use a QAManager properties file that includes the following line:

```
CONNECT_PARAMS=eng=MyServer;dbn=qanywhere
```

You can then start the MobiLink synchronization server just as you would

---

for any other QAnywhere messaging application, but use your MobiLink consolidated database as the server message store. For example:

```
dbmlsrv9 -m -c "dbf=your_consolidated.db"
```

## Setting up a failover mechanism

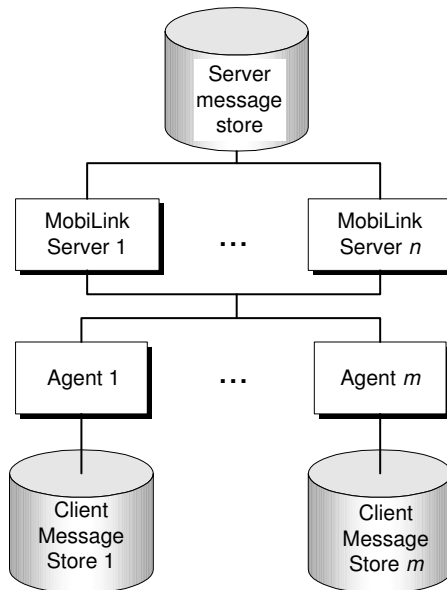
QAnywhere applications can be set up with failover mechanisms, such that there are alternate MobiLink synchronization servers that can be used if one fails. In order to support failover, each QAnywhere Agent must be started with a list of MobiLink servers. The first MobiLink server specified in the list is the primary server. The remaining servers in the list are alternate servers.

For example, running the following command on the remote device will start the QAnywhere Agent with one primary server and one alternate server:

```
qaagent -x tcpip(host=m11.ianywhere.com)
        -x tcpip(host=m12.ianywhere.com)
```

Each QAnywhere Agent can have a different primary server.

The following diagram describes a failover configuration in which you have multiple MobiLink servers and multiple QAnywhere agents. You have multiple client message stores, but all MobiLink servers are connected to the same server-side message store.



This configuration has the following characteristics:

- ◆ When a message transmission occurs, all messages in the server message store will be delivered to the client message store regardless of the server that the QAnywhere Agent is connected to.

- 
- ◆ Push Notifications will be sent to a QAnywhere Agent only when the QAnywhere Agent is connected to its primary server.
  - ◆ There is a single point of failure. If the machine with the server message store is unavailable, no messaging can take place.

---

## CHAPTER 4

# Writing QAnywhere Client Applications

About this chapter

This chapter describes how to use the QAnywhere API. The API is provided both as a C++ API and as the `iAnywhere.QAnywhere.Client` namespace.

Contents

| <b>Topic:</b>                                                          | <b>page</b> |
|------------------------------------------------------------------------|-------------|
| <a href="#">Introduction</a>                                           | 56          |
| <a href="#">Overview of writing a client application</a>               | 58          |
| <a href="#">Understanding QAnywhere message addresses</a>              | 59          |
| <a href="#">Initializing the QAnywhere client API</a>                  | 60          |
| <a href="#">Setting QAManager properties</a>                           | 64          |
| <a href="#">Sending QAnywhere messages</a>                             | 67          |
| <a href="#">Receiving messages synchronously</a>                       | 69          |
| <a href="#">Receiving messages asynchronously</a>                      | 70          |
| <a href="#">Reading very large messages</a>                            | 72          |
| <a href="#">Handling push notifications and network status changes</a> | 73          |
| <a href="#">Implementing transactional messaging</a>                   | 75          |
| <a href="#">Shutting down QAnywhere</a>                                | 77          |
| <a href="#">Deploying QAnywhere applications</a>                       | 78          |

---

## Introduction

QAnywhere client applications manage the receiving and sending of QAnywhere messages. The applications can be written using one of the following programming interfaces:

- ◆ **QAnywhere .NET API** A programming interface for deployment to Windows computers using the Microsoft .NET Framework and to handheld devices running the Microsoft .NET Compact Framework. The QAnywhere .NET API is provided as the `iAnywhere.QAnywhere.Client` namespace.

QAnywhere supports Microsoft Visual Studio 2003.

In this chapter, code samples for the .NET API use the C# programming language, but the API can be used from any programming language supported by Microsoft .NET.

Versions of the `TestMessage` sample application written in C#, Visual Basic .NET, and managed C++ are all supplied. For more information, see [“Lesson 5: Explore the TestMessage client source code” on page 22](#).

☞ For more information about the QAnywhere .NET API, see [“iAnywhere.QAnywhere.Client namespace” on page 173](#).

- ◆ **QAnywhere C++ API** A programming interface for deployment to Windows computers. This interface is for programmers using C++.

QAnywhere supports Microsoft Visual C++ 6.0, Microsoft Visual Studio .NET 2003, Microsoft eMbedded Visual C++ 3.0, and Microsoft eMbedded Visual C++ 4.0.

The QAnywhere C++ API consists of the following:

- A set of header files (the main one being `qa.hpp`) located in the `QAnywhere\h` subdirectory of your SQL Anywhere Studio installation.
- An import library (`qany9.lib`) located in the `QAnywhere\ce\arm.30\lib`, `QAnywhere\lib`, and `QAnywhere\ce\x86.30\lib` subdirectories of your SQL Anywhere Studio installation.
- A run-time DLL (`qany9.dll`) located in the `win32`, `ce\arm.30`, and `ce\x86.30` subdirectories of your SQL Anywhere Studio installation.

Your source code file must include the header file in order to access the API. The import library is used to link your application to the run-time DLL. The run-time DLL must be deployed with your application.

A version of the `TestMessage` sample application written in C++ is supplied in the `Samples\QAnywhere` subdirectory of your SQL Anywhere Studio installation. For more information, see [“Lesson 5: Explore the TestMessage client source code” on page 22](#).

☞ For more information about the C++ API, see [“QAnywhere C++ API Reference”](#) on page 121.

---

# Overview of writing a client application

## ❖ To build a client application (C++ or .NET)

1. Initialize the QAnywhere client API.
  - ☞ See [“Initializing the QAnywhere client API”](#) on page 60.
2. Set QAManager properties.
  - ☞ See [“Setting QAManager properties”](#) on page 64.
3. Write application code and compile. See:
  - ◆ [“Sending QAnywhere messages”](#) on page 67
  - ◆ [“Receiving messages synchronously”](#) on page 69
  - ◆ [“Receiving messages asynchronously”](#) on page 70
  - ◆ [“Reading very large messages”](#) on page 72
  - ◆ [“Handling push notifications and network status changes”](#) on page 73
  - ◆ [“Implementing transactional messaging”](#) on page 75
  - ◆ [“Shutting down QAnywhere”](#) on page 77
4. Deploy the application to the target device.
  - ☞ See [“Deploying QAnywhere applications”](#) on page 78.



## Understanding QAnywhere message addresses

A QAnywhere message destination has two parts:

- ◆ The client message store ID.

☞ For information about client message store IDs, see “[Setting up the client message store](#)” on page 35.

- ◆ The application queue name.

The queue name is specified inside the application, and must be known to instances of the sending application on other devices.

The form of the destination address is:

*id\queue-name*

### Escape characters and addresses

When constructing addresses as strings in an application, be sure to escape the backslash character if necessary. Follow the string escaping rules for the programming language you are using.

Sending a message to a JMS connector

A QAnywhere-to-JMS destination address has two parts:

- ◆ The connector address. This is the value of the `ianywhere.connector.address` property in the connector properties file.

☞ For more information, see “[Configuring the JMS connector properties file](#)” on page 43.

- ◆ The JMS queue name. This is a queue that you create using your JMS administration tools.

☞ For more information, see “[Using JMS Connectors](#)” on page 42.

The form of the destination address is:

*connector-address\JMS-queue-name*

☞ For more information about addressing messages in a JMS application, see “[Addressing QAnywhere messages meant for JMS](#)” on page 46 and “[Addressing JMS messages meant for QAnywhere](#)” on page 48.

---

# Initializing the QAnywhere client API

Before you can send or receive messages using QAnywhere, you must complete the following initialization tasks.

Setting up C++ applications

## ❖ To initialize the QAnywhere client API (C++)

1. Include the QAnywhere header file.

```
#include <qa.hpp>
```

*qa.hpp* defines the QAnywhere classes.

2. Initialize QAnywhere.

To do this, initialize a factory for creating QAManager objects.

```
QAManagerFactory *    factory;

factory = QAnywhereFactory_init();
if( factory == NULL ) {
    // fatal error
}
```

3. Create a QAManager object.

You can create a default QAManager object as follows:

```
QAManager *    mgr;

// Create a manager
mgr = factory->createQAManager( NULL );
if( mgr == NULL ) {
    // fatal error
}
```

You can customize a QAManager object programmatically or using a properties file.

- ◆ To customize QAManager programmatically, use the `setProperty` method.
  - ☞ For more information, see [“Setting properties programmatically” on page 65](#).
- ◆ To use a properties file, specify the properties file in the `createQAManager` method:

```
mgr = factory->createQAManager( "qa_mgr.props" );
```

where *qa\_mgr.props* is the name of the properties file on the remote device.

- ☞ For more information, see [“Setting properties in a file” on page 64](#).

## 4. Initialize the QAManager object.

```
qa_bool rc;
rc=mgr->open(
    AcknowledgementMode::IMPLICIT_ACKNOWLEDGEMENT ) )
```

The argument to the open method is an acknowledgement mode, which indicates how messages are to be acknowledged. It must be one of **IMPLICIT\_ACKNOWLEDGEMENT** or **EXPLICIT\_ACKNOWLEDGEMENT**. With implicit acknowledgement, messages are acknowledged as soon as they are received by the client. With explicit acknowledgement, you must call the acknowledgement method on the QAManager to acknowledge the message.

☞ For more information, see “Class QAManager” on page 138.

## Setting up .NET applications

You must make two changes to your Visual Studio .NET project to be able to use it:

- ◆ Add a reference to the QAnywhere .NET DLL. Adding a reference tells Visual Studio.NET which DLL to include to find the code for the QAnywhere .NET API.
- ◆ Add a line to your source code to reference the QAnywhere .NET API classes. In order to use the QAnywhere .NET API, you must add a line to your source code to reference the data provider. You must add a different line for C# than for Visual Basic.NET.

In addition, you must initialize the QAnywhere client API.

#### ❖ To add a reference to the QAnywhere .NET API in a Visual Studio .NET project

1. Start Visual Studio .NET and open your project.
2. In the Solution Explorer window, right-click the References folder and choose Add Reference from the popup menu.

The Add Reference dialog appears.

3. On the .NET tab, click Browse to locate iAnywhere.QAnywhere.Client.dll. (The default location is `|ProgramFiles|Sybase|SQLAnywhere9|win32`). Select the DLL and click Open.

Note that there is a separate version of the DLL for each of Windows and WindowsCE.

- 
4. You can verify that the DLL is added to your project. Open the Add Reference dialog and then click the .NET tab. `iAnywhere.QAnywhere.Client.dll` appears in the Selected Components list. Click OK to close the dialog.

The DLL is added to the References folder in the Solution Explorer window of your project. Referencing the data provider classes in your source code.

#### ❖ To reference the QAnywhere .NET API classes in your code

1. Start Visual Studio .NET and open your project.
2. If you are using C#, add the following line to the list of using directives at the beginning of your project:

```
using iAnywhere.QAnywhere.Client;
```

3. If you are using Visual Basic .NET, add the following line at the beginning of your project before the line `Public Class Form1`:

```
Imports iAnywhere.QAnywhere.Client
```

This line is not strictly required. However, it allows you to use short forms for the QAnywhere classes. Without it, you can still use

```
iAnywhere.QAnywhere.Client.QAManager  
mgr =  
    new iAnywhere.QAnywhere.Client.QAManagerFactory.Instance.C  
        reateQAManager(  
            "qa_manager.props" );
```

instead of

```
QAManager mgr = QAManagerFactory.Instance.CreateQAManager(  
    "qa_manager.props" );
```

in your code.

#### ❖ To initialize the QAnywhere client API (.NET)

1. Include the `iAnywhere.QAnywhere.Client` namespace.

```
using iAnywhere.QAnywhere.Client;
```

2. Create a QAManager object.

You can create a default QAManager object as follows:

```
QAManager mgr;  
mgr = QAManagerFactory.Instance.CreateQAManager( null );
```

You can alternatively create a QAManager object that is customized using a properties file. The properties file is specified in the CreateQAManager method:

```
mgr = QAManagerFactory.Instance.CreateQAManager(
    "qa_mgr.props" );
```

where *qa\_manager.props* is the name of the properties file that resides on the remote device.

3. Initialize the QAManager object.

```
mgr.Open(
    AcknowledgementMode.EXPLICIT_ACKNOWLEDGEMENT);
```

The argument to the open method is an acknowledgement mode, which indicates how messages are to be acknowledged. It must be one of **IMPLICIT\_ACKNOWLEDGEMENT** or **EXPLICIT\_ACKNOWLEDGEMENT**. With implicit acknowledgement, messages are acknowledged as soon as they are received by the client. With explicit acknowledgement, you must call the acknowledgement method on the QAManager to acknowledge the message.

☞ For more information, see [“QAManager class” on page 195](#).

You are now ready to send messages.

---

## Setting QAManager properties

You have two options for setting QAManager properties:

- ◆ Create a properties text file to define the QAnywhere Manager properties that will be used by one Manager instance.
- ◆ Programmatically set QAnywhere Manager properties.

### Setting properties in a file

The information in a QAManager properties file is specific to one instance of a QAManager. An instance of QAManager can only be used by the thread that creates it. Applications may use more than one QAManager, but each QAManager cannot be shared by more than one thread.

When using a properties file, it must be configured for and installed on the remote device with each deployed copy of your application.

You specify the name of this file in your application. If the properties file does not reside in the same directory as your client executable, you must also specify the absolute path. If you want to use the default settings for the properties, use NULL instead of a filename.

Values set in the file permit you to enable or disable some of the QAnywhere features, such as automatic message compression and logging.

☞ For more information, see [“QAManager properties” on page 66](#).

Lines of this file that begin with the character # are treated as comments.

#### Example

For example, suppose you have a QAnywhere Manager properties file called *mymanager.props* with the following content:

```
COMPRESSION_LEVEL=9
CONNECT_PARMS=DBF=mystore.db
```

When you create QAManager, you reference the file by name.

Following is an example using C++:

```
QAManagerFactory * qa_factory;
QAManager * mgr;

qa_factory = QAnywhereFactory_init();
qa_factory->createQAManager( "mymanager.props" );
mgr->open( AcknowledgementMode::EXPLICIT_ACKNOWLEDGEMENT );
```

Following is an example using C#:

```

QAManager      mgr;

mgr = QAManagerFactory.Instance.CreateQAManager(
    "mymanager.props" );
mgr.Open( AcknowledgeMode.EXPLICIT_ACKNOWLEDGEMENT );

```

## Setting properties programmatically

You can use the QAManager setProperties method to set properties programmatically. Setting QAManager properties programmatically must be done before calling the Open() method of a QAManager instance.

### Example

The following C++ example sets properties programmatically. When you create the QAManager, you specify the property settings.

```

QAManagerFactory *   qa_factory;
QAManager *          mgr;

qa_factory = QAnywhereFactory_init();
mgr->createQAManager( NULL );
mgr->setProperty( "COMPRESSION_LEVEL", "9" );
mgr->setProperty( "CONNECT_PARAMS", "DBF=mystore.db" );
mgr->open( AcknowledgeMode::EXPLICIT_ACKNOWLEDGEMENT );

```

The following C# example sets properties programmatically. When you create the QAManager, you specify the property settings.

```

QAManager      mgr;

mgr = QAManagerFactory.Instance.CreateQAManager( null );
mgr.SetProperty( "COMPRESSION_LEVEL", "9" );
mgr.SetProperty( "CONNECT_PARAMS", "DBF=mystore.db" );
mgr.Open( AcknowledgeMode.EXPLICIT_ACKNOWLEDGEMENT );

```

### See also

- ◆ For .NET: [“iAnywhere.QAnywhere.Client namespace” on page 173](#)
- ◆ For C++: [“QAnywhere C++ API Reference” on page 121](#)
- ◆ [“QAManager properties” on page 66](#)

---

## QAManager properties

The following table lists all the available properties.

| Property                                      | Notes                                                                                                                                                                                                                                                                             |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>COMPRESSION_LEVEL</b> = <i>n</i>           | Set the compression level.<br><i>n</i> is the compression factor. It is an integer between 0 and 9, 0 indicating no compression and 9 indicating maximum compression.                                                                                                             |
| <b>CONNECT_PARAMS</b> = <i>connect-string</i> | A connection string used by the QAnywhere Manager to connect to the message store database. The connect options are written in the form <i>keyword=value</i> . For a list of options, see <a href="#">“Connection parameters” [ASA Database Administration Guide, page 176]</a> . |
| <b>LOG_FILE</b> = <i>“filename”</i>           | The name of a file to which logging messages are to be written. Implicitly enables logging.                                                                                                                                                                                       |
| <b>MAX_IN_MEMORY_MESSAGE_SIZE</b> = <i>n</i>  | When reading a message, <i>n</i> is the largest message, in bytes, for which a buffer will be allocated. A message larger than <i>n</i> bytes must be read using streaming operations. The default is 1000000 on Windows and 64000 on Windows CE.                                 |



## Sending QAnywhere messages

The following procedures describe how to send messages from QAnywhere applications. These procedures assume that you have created and opened a QAManager object.

Sending a message from your application does *not* ensure it is delivered from your device. It simply places the message on a queue to be delivered. The QAnywhere Agent carries out the task of sending the message to the MobiLink synchronization server, which in turn delivers it to its destination.

### ❖ To send a message (C++)

1. Create a new message.

You can create either a text message or a binary message. Use the method for the type of message that you want to send.

```
QATextMessage *   msg;
msg = mgr->createTextMessage();
```

2. Set message properties.

Use methods of the QATextMessage or QABinaryMessage class to set properties.

3. Put the message on the queue, ready for sending.

```
if( msg != NULL ) {
    if( !mgr->putMessage( "store-id\\queue-name", msg ) )
    {
        // display error using mgr->getLastErrorMsg()
    }
    mgr->deleteMessage( msg );
}
```

where *store-id* and *queue-name* are strings that combine to form the destination address.

### ❖ To send a message (.NET)

1. Create a new message.

You can create either a text message or a binary message. Use the method for the type of message that you want to send.

```
QATextMessage     msg;
msg = mgr.CreateTextMessage();
```

2. Set message properties.

Use methods of the QATextMessage or QABinaryMessage class to set properties.

---

3. Put the message on the queue, ready for sending.

```
mgr.PutMessage( "store-id\\queue-name", msg );
```

where *store-id* and *queue-name* are strings that combine to form the destination address.

## Receiving messages synchronously

To receive messages synchronously, your application explicitly polls the queue for messages. It may poll the queue periodically, or when a user initiates a particular action such as clicking a Refresh button.

### ❖ To receive messages synchronously (C++)

1. Declare message objects to hold the incoming messages.

```
QAMessage *      msg;
QATextMessage *  text_msg;
```

2. Poll the message queue, receiving messages:

```
if( mgr->start() ) {
    for( ;; ) {
        msg = mgr->getMessageNoWait( "queue_name" );
        if( msg == NULL ) break;
        text_msg = msg->castToTextMessage();
        if( text_msg != NULL ) {
            // display text message using
            // text_msg->getText()
        }
        sleep( time-period );
    }
    mgr->stop();
}
```

### ❖ To receive messages synchronously (.NET)

1. Declare message objects to hold the incoming messages.

```
QAMessage      msg;
QATextMessage  text_msg;
```

2. Poll the message queue, collecting messages:

```
if(mgr.start() ) {
    for(;;) {
        msg = mgr.GetMessageNoWait("queue_name");
        if( msg == null ) break;
        addMessage( msg );
    }
    mgr.stop();
}
```

---

## Receiving messages asynchronously

To receive messages asynchronously you must write and register a message listener function that is called by QAnywhere when a message appears in the queue. The message listener takes the incoming message as a parameter. The task you perform in your message listener depends on your application. For example, in the TestMessage sample application that is installed with SQL Anywhere Studio, the message listener adds the message to the list of messages in the main TestMessage window.

### ❖ To receive messages asynchronously (C++)

1. Create a class that implements the QAMessageListener interface.

```
class MyClass: public QAMessageListener
{
private:
    void onMessage( QAMessage * Msg);
};
```

2. Implement the onMessage method.

The QAMessageListener interface contains one method, onMessage. Each time a message arrives in the queue, the QAnywhere library calls this method, passing the new message as the single argument.

```
void MyClass::onMessage(QAMessage * msg)
{
    // process msg
}
```

Be particularly careful to handle errors. Failure to do so can cause you to miss messages.

3. Register the message listener.

```
my_listener = new MyClass();
mgr->setMessageListener( "queue-name", my_listener );
```

### ❖ To receive messages asynchronously (.NET)

1. Implement a message handler.

```
private void onMessage(QAMessage msg)
{
    // process msg
}
```

2. Register the message handler.

To register a message handler, create a `QAManager.MessageListener` object that has the message handler function as its argument. Then use the `QAManager.SetMessageListener` function to register the `MessageListener` with a specific queue.

```
QAManager.MessageListener listener;  
listener=new QAManager.MessageListener( onMessage );  
mgr.SetMessageListener( "queue-name", listener );
```

where *queue-name* is a string and is the name of the queue the `QAManager` object listens to.

---

## Reading very large messages

Sometimes messages are so large that they exceed the limit set with the QAManager property `MAX_IN_MEMORY_MESSAGE_SIZE` or its defaults of 1MB on Windows and 64KB on Windows CE. In this case the message cannot be read from memory, so the usual read methods such as `readInt` and `readString` cannot be used. However, you can read very large messages directly from the message store in pieces. To do this, use `QATextMessage.readText` or `QABinaryMessage.readBinary` in a loop. At the end of the read, -1 is returned.

When you do this, you cannot use a QAManager that was opened with `IMPLICIT_ACKNOWLEDGEMENT`. You must use a QAManager that was opened with `EXPLICIT_ACKNOWLEDGEMENT` and you must do all calls to `readText` or `readBinary` before acknowledging the message.

## Handling push notifications and network status changes

☞ For background information about using notifications, see [“Scenario for messaging with push notifications”](#) on page 6.

Notifications and network status changes are both sent to QAnywhere applications as **system messages**. System messages are exactly the same as other messages, but are received in a separate queue named **system**.

For example, the following C# code deals with system and normal messages. It assumes that you have defined the message handling functions `onMessage` and `onSystemMessage` that implement the application logic for processing the messages.

```
// Declare the message listener and system listener
private QAManager.MessageListener _receiveListener;
private QAManager.MessageListener _systemListener;
...

// Create a MessageListener that uses the appropriate message
// handlers
_receiveListener = new QAManager.MessageListener( onMessage );
_systemListener = new QAManager.MessageListener( onSystemMessage );
...

// Register the message handler
mgr.SetMessageListener( queue-name, _receiveListener );
mgr.SetMessageListener( "system", _systemListener );
```

The system message handler may query the message properties to identify what information it contains. The message type property indicates if the message holds a network status notification. For example, for a message `msg`:

```
if( msg.PropertyExists(MessageProperties.MSG_TYPE) ) {
    msg_type=(MessageType)msg.GetIntProperty(...);
    ...
} else {
    // Process regular message
}
```

---

```
msg_type = (MessageType)msg.GetIntProperty(
    MessageProperties.MSG_TYPE );
if( msg_type == MessageType.NETWORK_STATUS_NOTIFICATION ) {
    // Process a network status change
} else if ( msg_type == MessageType.PUSH_NOTIFICATION ) {
    // Process a push notification
} else if ( msg_type == MessageType.REGULAR ) {
    // This message type should not be received on the system
    // queue. Take appropriate action here.
}
```

See also

- ◆ [ias\\_Status](#) in “[Message headers](#)” on page 110
- ◆ [ias\\_Network](#) in “[Client store properties](#)” on page 113



## Implementing transactional messaging

Transactional messaging provides the ability to group messages in a way that guarantees that either all messages in the group are delivered, or none are. This is more commonly referred to as a single **transaction**.

When implementing transactional messaging, you create a special QAManager object called a transactional manager.

### ❖ To create a transactional manager (C++)

1. Initialize QAnywhere.

This step is exactly the same as in non-transactional messaging.

```
#include <qa.hpp>
QAManagerFactory * factory;

factory = QAnywhereFactory_init();
if( factory == NULL ) {
    // fatal error
}
```

2. Create a transactional manager.

```
QATransactionalManager * mgr;

mgr = factory->createQATransactionalManager( NULL );
if( mgr == NULL ) {
    // fatal error
}
```

As with non-transactional managers, you can specify a properties file to customize QAnywhere behavior. In this example, no properties file is used.

3. Initialize the manager.

```
if( !mgr->open() ) {
    // display message using mgr->getLastErrorMsg();
}
```

You are now ready to send messages. The following procedure sends two messages in a single transaction.

### ❖ To send multiple messages in a single transaction (C++)

1. Initialize message objects.

```
QATextMessage * msg_1;
QATextMessage * msg_2;
```

---

## 2. Send the messages.

The following code sends two messages in a single transaction:

```
msg_1 = mgr->createTextMessage();
if( msg_1 != NULL )
{
    msg_2 = mgr->createTextMessage();
    if( msg_2 != NULL )
    {
        if( !mgr->putMessage( "jms_1\\queue_name", msg_1 ) )
        {
            // display message using mgr->getLastErrorMsg();
        } else {
            if( !mgr->putMessage( "jms_1\\queue_name", msg_2 ) )
            {
                // display message using mgr->getLastErrorMsg();
            } else {
                mgr->commit();
            }
        }
        mgr->deleteMessage( msg_2 );
    }
    mgr->deleteMessage( msg_1 );
}
```

The `commit()` method actually sends the messages.

## Shutting down QAnywhere

After you have completed sending and receiving messages, you can shut down the QAnywhere messaging system by completing the following step.

### ❖ To shut down QAnywhere (C++)

1. Close the QAManager.

```
mgr->stop();  
mgr->close();
```

2. Terminate the factory.

```
QAnywhereFactory_term();
```

This step shuts down the messaging part of your application.

### ❖ To shut down QAnywhere (.NET)

1. Stop and close the QAManager.

```
mgr.Stop();  
mgr.Close();
```

---

## Deploying QAnywhere applications

☞ For information about the files needed to deploy QAnywhere applications, see “[Deploying QAnywhere applications](#)” [*MobiLink Administration Guide*, page 561].

---

## CHAPTER 5

# QAnywhere Agent

About this chapter

This chapter describes the syntax of the QAnywhere Agent, qaagent.

☞ For an overview of how to run the QAnywhere Agent, see [“Running the QAnywhere Agent”](#) on page 37.

Contents

| <b>Topic:</b>                          | <b>page</b> |
|----------------------------------------|-------------|
| <a href="#">QAnywhere Agent syntax</a> | 80          |

---

# QAnywhere Agent syntax

Use the QAnywhere Agent to send and receive messages for all QAnywhere applications on a single client device.

## Usage

The QAnywhere Agent controls other processes, which handle various messaging tasks. When you start the QAnywhere Agent command it spawns the following processes:

- ◆ **dbmlsync** The dbmlsync executable is the MobiLink synchronization client. MobiLink synchronization is used to send and receive messages, so the dbmlsync executable is required.
- ◆ **dblsn** The dblsn executable is the Listener utility. It receives push notifications. If you are not using push notifications, you do not need to supply the dblsn executable when you deploy your application, and you must run qaagent with push\_notifications disabled.
- ◆ **database server** The client message store is an Adaptive Server Anywhere database. QAnywhere Agent requires the Adaptive Server Anywhere database server to run the database. For Windows CE, the database server is *dbsrv9.exe*. For Windows, the database server is the personal database server *dbeng9.exe*.

The QAnywhere Agent can spawn a database server or connect to a running server, depending on the communication parameters that you specify in the qaagent -c option.

Each of these processes is managed by the QAnywhere Agent, and does not need to be managed separately.

☞ For deployment information, see “[Deploying QAnywhere applications](#)” on page 78.

## Syntax

**qaagent** [ *option ...* ]

| Option                      | Description                                                                                                                          |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| @ <i>data</i>               | Reads options from the specified environment variable or configuration file. See “ <a href="#">@data option</a> ” on page 82.        |
| -c <i>connection-string</i> | Specify a connection string to the client message store. See “ <a href="#">-c option</a> ” on page 82.                               |
| -id <i>id</i>               | Specify the ID of the client message store that the QAnywhere Agent is to connect to. See “ <a href="#">-id option</a> ” on page 84. |

| Option                                     | Description                                                                                                                                                                                        |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>-iu</b> <i>upload-size</i>              | Controls the size, in bytes, of incremental uploads. Use k or m to specify units of kilobytes or megabytes, respectively. See “ <a href="#">-iu option</a> ” on page 85.                           |
| <b>-la_port</b> <i>number</i>              | Specify the port on which the Listener listens for notifications from the MobiLink synchronization server. The default is 5001. See “ <a href="#">-la_port option</a> ” on page 85.                |
| <b>-mp</b> <i>password</i>                 | Specify the MobiLink password for the ID being synchronized. See “ <a href="#">-mp option</a> ” on page 85.                                                                                        |
| <b>-o</b> <i>logfile</i>                   | Log output messages to this file. See “ <a href="#">-o option</a> ” on page 86.                                                                                                                    |
| <b>-on</b> <i>size</i>                     | Specify a maximum size for the QAnywhere Agent message log file, after which the file is renamed with the extension .old and a new file is started. See “ <a href="#">-on option</a> ” on page 86. |
| <b>-os</b> <i>size</i>                     | Specifies a maximum size for the QAnywhere Agent message log file, after which a new log file with a new name is created and used. See “ <a href="#">-os option</a> ” on page 87.                  |
| <b>-ot</b> <i>logfile</i>                  | Truncate file and log output messages to file. See “ <a href="#">-ot option</a> ” on page 87.                                                                                                      |
| <b>-policy</b> <i>policy-type</i>          | Specify the transmission policy used by the QAnywhere Agent. See “ <a href="#">-policy option</a> ” on page 88.                                                                                    |
| <b>-port</b> <i>number</i>                 | Specify the port on which to listen for messages from the Listener. The default is 5002. See “ <a href="#">-port option</a> ” on page 90.                                                          |
| <b>-push_notifications</b><br><i>value</i> | Enable or disable push notifications. The default is enabled. See “ <a href="#">-push_notifications option</a> ” on page 90.                                                                       |
| <b>-q</b>                                  | Start the QAnywhere Agent in quiet mode with the window minimized in the system tray. See “ <a href="#">-q option</a> ” on page 90.                                                                |
| <b>-qi</b>                                 | Start the QAnywhere Agent in quiet mode with the window completely hidden. See “ <a href="#">-qi option</a> ” on page 91.                                                                          |

| Option                                                         | Description                                                                                                                             |
|----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>-si</b>                                                     | Initialize the database for use as a client message store. See “ <a href="#">-si option</a> ” on page 91.                               |
| <b>-su</b>                                                     | Upgrades a client message store from SQL Anywhere Studio version 9.0.1 to version 9.0.2. See “ <a href="#">-su option</a> ” on page 92. |
| <b>-v</b> [ <i>levels</i> ]                                    | Specify the level of verbosity. See “ <a href="#">-v option</a> ” on page 92.                                                           |
| <b>-x</b> { <b>http tcpip</b> } [ <i>(keyword=value;...)</i> ] | Specify protocol options for communication with the MobiLink synchronization server. See “ <a href="#">-x option</a> ” on page 93.      |

## @data option

Function Reads options from the specified environment variable or configuration file.

Syntax **qaagent** @{ *filename* | *environment-variable* } ...

Remarks With this option, you can put command line options in an environment variable or configuration file. If both exist with the name you specify, the environment variable is used.

☞ For more information about configuration files, see “[Using configuration files](#)” [*ASA Database Administration Guide*, page 495].

If you want to protect passwords or other information in the configuration file, you can use the File Hiding utility to obfuscate the contents of the configuration file.

☞ See “[Hiding the contents of files using the dbfhide command-line utility](#)” [*ASA Database Administration Guide*, page 524].

## -c option

Function Specify a connection string to the client message store.

Syntax **qaagent -c** *connection-string* ...

Defaults

- ◆ uid - **ml\_qa\_user**
- ◆ pwd - **qanywhere**
- ◆ dbn - **id** (where *id* is the client message store ID, and is specified with the **-id** option or defaults to the device name)
- ◆ dbf - **id.db** (where *id* is the client message store ID, and is specified with the **-id** option or defaults to the device name)



## Remarks

The connection string must specify connection parameters in the form *keyword=value*, separated by semicolons, with no spaces between parameters.

Connection parameters must be included in the ODBC data source specification if not given in the command line. Check your RDBMS and ODBC data source to determine required connection data.

☞ For a complete list of connection parameters, see “[Connection parameters](#)” [*ASA Database Administration Guide*, page 176].

Following are some of the connection parameters you may need to use:

- ◆ **dbf=filename** If you want QAnywhere Agent to start the client message store (rather than starting it from another application), you must specify a database file name. The default value of the database file is the name of the database. The default name for the database is the device name followed by *.db*; or you can specify a name with the *-dbn* connection parameter.

- ◆ **dbn=database-name** If the client message store is already running when the QAnywhere Agent starts, you can connect to it by specifying a database name rather than a database file. The QAnywhere Agent looks on the default Adaptive Server Anywhere database server for a database with this name.

The default database name of the client message store is the ID; for example, *agent1*. You can specify a name for the ID using the *qaagent -id* option, or use the default client message store ID, which is the device name.

- ◆ **eng=server-name** If you want to use a database server that is already running, use this option to specify the server name. The default value is the name of the database.

- ◆ **uid=user** Specify a database user ID to connect to the client message store. This is required if you change the defaults.

- ◆ **pwd=password** Specify the password for the database user ID. This is required if you change the defaults.

- ◆ **dbkey=key** If the client message store is encrypted using strong encryption, specify the encryption key required to access the database.

## See also

- ◆ “[Connection parameters](#)” [*ASA Database Administration Guide*, page 176]
- ◆ “[Connecting to a Database](#)” [*ASA Database Administration Guide*, page 37]
- ◆ “[Introduction to iAnywhere Solutions ODBC Drivers](#)” [*ODBC Drivers for MobiLink and Remote Data Access*, page 1]

---

Example

```
qaagent -id Device1 -c "DBF=qanyclient.db" -x  
tcpip(host=hostname) -policy automatic
```

## -id option

Function

Specify the ID of the client message store that the QAnywhere Agent is to connect to.

Syntax

**qaagent -id id ...**

Default

The default value of the ID is the machine name on which the Agent is running. In some cases, machine names may not be unique, in which case you must use the -id option.

Remarks

Each client message store is represented by a unique sequence of characters called the message store ID. If you do not supply an ID when you first connect to the message store, the default is the device name. On subsequent connections, you must always specify the same message store ID with the -id option.

The message store ID is a MobiLink user name, and it is required because in all MobiLink applications, each remote database must have a unique MobiLink user name. As with all MobiLink user names, this must be added to a MobiLink system table on the consolidated database. You must do this even if you use the default, unless you use custom authentication.

☞ For more information, see [“Creating MobiLink users”](#) [*MobiLink Clients*, page 10].

You cannot use the following characters in an ID:

- ◆ double quotes
- ◆ control characters
- ◆ double backslashes

In addition,

- ◆ You can use a single backslash only if it is used as an escape character.
- ◆ If your client message store database has the QUOTED\_IDENTIFIER database option set to off (not the default), then your ID can only include alphanumeric characters and underscores, at signs, pounds, and dollar signs.

See also

- ◆ [“About MobiLink users”](#) [*MobiLink Clients*, page 10]
- ◆ [“Setting up the client message store”](#) on page 35

## -iu option

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function | Synchronize messages using the MobiLink incremental uploads feature.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Syntax   | <b>qaagent -iu</b> <i>upload-size</i> [ <b>k</b>   <b>m</b> ] ...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Default  | Uploads are sent as a single unit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Remarks  | <p>Using -iu, QAnywhere can synchronize messages using the MobiLink incremental uploads feature. This option specifies a minimum incremental scan volume in units of bytes.</p> <p>Use the suffix k or m to specify units of kilobytes or megabytes, respectively.</p> <p>When this option is specified, uploads are sent to the MobiLink synchronization server in one or more parts. This could be useful if a site has difficulty maintaining a connection for long enough to complete the full upload. When the option is not set, uploads are sent as a single unit.</p> <p>Incremental uploads may be less efficient in that they send the schema with each increment.</p> |
| See also | ◆ <a href="#">“Increment (inc) extended option”</a> [ <i>MobiLink Clients</i> , page 121]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## -la\_port option

|          |                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function | Specify the Listener port.                                                                                                                                                                                                                                                                                                                                                                               |
| Syntax   | <b>qaagent -la_port</b> <i>number</i> ...                                                                                                                                                                                                                                                                                                                                                                |
| Default  | <b>5001</b>                                                                                                                                                                                                                                                                                                                                                                                              |
| Remarks  | <p>The port number on which the Listener listens for notifications from the MobiLink synchronization server. Notifications are used to inform the QAnywhere Agent that a message is waiting.</p> <p>If no Listener is running on the device, qaagent starts the Listener on the specified port. If a Listener is running, you must specify the port that it is running on (or use the default port).</p> |
| See also | <ul style="list-style-type: none"> <li>◆ <a href="#">“Scenario for messaging with push notifications”</a> on page 6</li> <li>◆ <a href="#">“-push_notifications option”</a> on page 90</li> </ul>                                                                                                                                                                                                        |

## -mp option

|          |                                                                |
|----------|----------------------------------------------------------------|
| Function | Specify the MobiLink password for the client message store ID. |
| Syntax   | <b>qaagent -mp</b> <i>password</i> ...                         |
| Default  | None                                                           |

---

Remarks If the MobiLink synchronization server requires user authentication, use `-mp` to supply the MobiLink password corresponding to the ID. The ID identifies the client message store and is specified with the `-id` option.

See also

- ◆ [“Authenticating MobiLink Users”](#) [*MobiLink Clients*, page 9]
- ◆ [“-id option”](#) on page 84

## **-o option**

Function Sends output to a log file.

Syntax **qaagent -o logfile ...**

Default None

Remarks The QAnywhere Agent logs output to the file name that you specify. The Adaptive Server Anywhere synchronization client (`dbmlsync`) logs output to a file with the same name, but including the suffix `_sync`. The Listener utility (`dblsn`) logs output to a file with the same name, but including the suffix `_lsn`.

For example, if you specify the log file `c:\tmp\mylog.out`, then `qaagent` will log to `c:\tmp\mylog.out`, `dbmlsync` will log to `c:\tmp\mylog_sync.out` and `dblsn` will log to `c:\tmp\mylog_lsn.out`.

See also

- ◆ [“-ot option”](#) on page 87
- ◆ [“-on option”](#) on page 86
- ◆ [“-os option”](#) on page 87
- ◆ [“-v option”](#) on page 92

## **-on option**

Function Specifies a maximum size for the QAnywhere Agent message log file, after which the file is renamed with the extension `.old` and a new file is started.

Syntax **qaagent -on size [ k | m ] . . .**

Default None

Description The `size` is the maximum file size for the output log, in bytes. Use the suffix `k` or `m` to specify units of kilobytes or megabytes, respectively. The minimum size limit is 10KB.

When the log file reaches the specified size, the QAnywhere Agent renames the output file with the extension `.old`, and starts a new one with the original name.

**Note**

If the *.old* file already exists, it is overwritten. To avoid losing old log files, use the `-os` option instead.

This option cannot be used with the `-os` option.

- See also
- ◆ [“-o option” on page 86](#)
  - ◆ [“-ot option” on page 87](#)
  - ◆ [“-os option” on page 87](#)
  - ◆ [“-v option” on page 92](#)

**-os option**

- Function** Specifies a maximum size for the QAnywhere Agent message log file, after which a new log file with a new name is created and used.
- Syntax** **dbmlsrv9 -os** *size* [ **k** | **m** ] . . .
- Default** None
- Description** The *size* is the maximum file size for logging output messages. The default units is bytes. Use the suffix *k* or *m* to specify units of kilobytes or megabytes, respectively. The minimum size limit is 10KB.
- Before the QAnywhere Agent logs output messages to a file, it checks the current file size. If the log message will make the file size exceed the specified size, the QAnywhere Agent renames the message log file to *yymmddxx.mls*. In this instance, *xx* are sequential characters ranging from 00 to 99, and *yymmdd* represents the current year, month, and day.
- You can use this option to prune old message log files to free up disk space. The latest output is always appended to the file specified by `-o` or `-ot`.
- You cannot use this option with the `-on` option.

- See also
- ◆ [“-o option” on page 86](#)
  - ◆ [“-ot option” on page 87](#)
  - ◆ [“-on option” on page 86](#)
  - ◆ [“-v option” on page 92](#)

**-ot option**

- Function** Truncates the log file and appends output messages to it.
- Syntax** **qaagent -ot** *logfile* ...
- Default** None
- Remarks** The QAnywhere Agent logs output to the file name that you specify. The

---

Adaptive Server Anywhere synchronization client (dbmsync) logs output to a file with the same name, but including the suffix `_sync`. The Listener utility (dblsn) logs output to a file with the same name, but including the suffix `_lsn`.

For example, if you specify the log file `c:\tmp\mylog.out`, then `qaagent` will log to `c:\tmp\mylog.out`, `dbmsync` will log to `c:\tmp\mylog_sync.out` and `dblsn` will log to `c:\tmp\mylog_lsn.out`.

See also

- ◆ “-o option” on page 86
- ◆ “-on option” on page 86
- ◆ “-os option” on page 87
- ◆ “-v option” on page 92

## -policy option

Function

Specify a policy to determine when to send and receive messages.

Syntax

**qaagent -policy** *policy-type* ...

*policy-type*: **ondemand** | **scheduled**[ *interval* ] | **automatic** | *rules-file*

Defaults

- ◆ Scheduled policy with transmissions every 10 seconds.
- ◆ If you do not specify persistence rules in a transmission rules file, messages are deleted when the final status of the message is determined to be received or expired.

Remarks

QAnywhere uses a policy to determine when to send and receive messages. The *policy-type* can be one of the following:

- ◆ **ondemand** Only send messages when the QAnywhere client application makes the appropriate function call.  
The `QAManager.PutMessage()` method causes messages to be queued locally. These messages are not transmitted to the server until the `QAManager.TriggerSendReceive()` method is called. Similarly, messages waiting on the server are not sent to the client until the `TriggerSendReceive()` method is called by the client.

You can use `ondemand` policy to respond to notifications, but must do so in your application. A notification causes a system message to be delivered to the QAnywhere client. In your application, you may choose to respond to this system message by calling `TriggerSendReceive()`.

☞ For information about notifications, see “[Scenario for messaging with push notifications](#)” on page 6.

- ◆ **scheduled** Send and receive messages at a specified interval. The default is 10 seconds.

Transmission of messages between the client and the server takes place at a specified time interval.

The `QAManager.PutMessage()` method causes messages to be queued locally. These messages are not transmitted until the time interval has elapsed. Messages queued on the server for delivery to the client are also transmitted when the time interval has elapsed.

If push notifications are enabled, messages queued on the server for delivery to the client are transmitted when the next time interval elapses.

The `TriggerSendReceive()` method can be used to override the time interval. It forces a message transmission to occur before the time interval elapses.

The optional *interval* argument is the number of seconds between send and receive operations. For example, the following command schedules the QA Agent to synchronize messages every 15 minutes:

```
qaagent.exe -policy scheduled[900]
```

- ◆ **automatic** Receive messages when a push notification is received or when one of the other events, described below, occurs.

The QAnywhere agent attempts to keep message queues as current as possible. Any of the following events cause messages queued on the client to be delivered to the server and messages queued on the server to be delivered to the client:

- Invoking the `PutMessage()` method.
- Invoking the `TriggerSendReceive()` method.
- A notification.
  - ☞ For information about notifications, see [“Scenario for messaging with push notifications”](#) on page 6.
- A message status change on the client. For example, a status change occurs when an application retrieves a message from a local queue which causes the message status to change from pending to delivered.

- ◆ **rules-file** Specify a transmission rules file. The transmission rules file can indicate a more complicated set of rules to determine when messages are sent.

☞ For more information about transmission rules, see [“Transmission rules”](#) on page 102.

See also

- ◆ [“Determining when message transmission should occur on the client”](#) on page 37
- ◆ [“QAnywhere C++ API Reference”](#) on page 121
- ◆ [“iAnywhere.QAnywhere.Client namespace”](#) on page 173

---

## -port option

|          |                                                                                                                                                                                                                                                            |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function | Specify a port for the QAnywhere Agent.                                                                                                                                                                                                                    |
| Syntax   | <b>qaagent -port</b> <i>number</i> ...                                                                                                                                                                                                                     |
| Default  | <b>5002</b>                                                                                                                                                                                                                                                |
| Remarks  | <p>The port number on which QAnywhere Agent listens for communications from the Listener.</p> <p>The default value is <b>5002</b>. If you set a different port, the QAnywhere Agent automatically configures the Listener to communicate on that port.</p> |

## -push\_notifications option

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function | Specify whether server notifications are supported.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Syntax   | <b>qaagent -push_notifications</b> { <b>enabled</b>   <b>disabled</b> } ...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Default  | Enabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Remarks  | <p>If you do not want to use notifications, set this option to disabled. You then do not have to deploy the <i>dblsn.exe</i> executable with your clients.</p> <p>☞ For a description of this setup, see <a href="#">“Simple messaging scenario” on page 5</a>.</p> <p>The <i>dblsn.exe</i> executable (the Listener) is not supported on Windows 95 or Windows NT.</p> <p>If you are using UDP, you cannot use push notifications with ActiveSync due to the limitations of the UDP implementation of ActiveSync.</p> <p>☞ To use push notifications with SMS, see <a href="#">“Using push notifications with SMS” on page 40</a>.</p> |

## -q option

|          |                                                                                                                                                                                                   |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function | Start the QAnywhere Agent in quiet mode with the window minimized in the system tray.                                                                                                             |
| Syntax   | <b>qaagent -q</b> ...                                                                                                                                                                             |
| Default  | None.                                                                                                                                                                                             |
| Remarks  | When you start the QAnywhere Agent in quiet mode with -q, the main window is minimized to the system tray. In addition, the database server for the message store is started with the -qi option. |



See also ♦ [“-qi option” on page 91](#)

## -qi option

Start the QAnywhere Agent in quiet mode with the window completely hidden.

Syntax **qaagent -qi ...**

Default None.

Remarks When you start the QAnywhere Agent in quiet mode, on Windows desktop the main window is minimized to the system tray, and on Windows CE the main window is hidden. In addition, the database server for the message store is started with the -qi option.

Quiet mode is useful for some Windows CE applications because it prevents the situation where the application is closed when Windows CE reaches its limit of 32 concurrent processes. Quiet mode allows the QAnywhere Agent to run like a service.

When in -qi quiet mode, you can only stop the QAnywhere Agent by running **qastop**.

See also ♦ [“-q option” on page 90](#)

## -si option

Function Initialize the database for use as a client message store.

Syntax **qaagent -c “*connection-string*” -si ...**

Default None. You only use this option once, to initialize the client message store.

Remarks Before using this option, you must create an Adaptive Server Anywhere database. This database should not be used for any purpose other than as a message store. When you use -si, the QAnywhere Agent initializes the database with database objects such as MobiLink system tables; it then exits immediately.

When you run -si, you must specify a connection string with the -c option that indicates which database to initialize. The connection string specified in the -c option should also specify a user ID with DBA privileges. If you do not specify a user ID and password, the default user DBA with password SQL is used.

The -si option creates a database user named **ml\_qa\_user** and password **qanywhere** for the client message store. The user called ml\_qa\_user has permissions suitable for QAnywhere applications only. If you do not change

---

this database user name and password, then you do not need to specify the `pwd` or `uid` in the `-c` option when you start `qaagent`. If you change either of them, then you must supply the `uid` and/or `pwd` in the `-c` option on the `qaagent` command line.

☞ You should change the default passwords. To change them, use the `GRANT` statement. For more information, see [“Changing a password” \[ASA Database Administration Guide, page 434\]](#).

The `-si` option does not provide an ID for the client message store. You can assign an ID using the `-id` option when you run `-si` or the next time you run `qaagent`; or, if you do not do that, `qaagent` will by default assign the device name as the ID.

When a message store is set up but does not have an ID, QAnywhere applications local to the message store can send and receive messages, but cannot exchange messages with remote QAnywhere applications. Once an ID is assigned, remote messaging may also occur.

See also

- ◆ [“Setting up the client message store” on page 35](#)
- ◆ [“Creating a secure client message store” on page 96](#)

Examples

The following command connects to a database called `qaclient.db` and initializes it as a QAnywhere client message store. The QAnywhere Agent immediately exits when the initialization is complete.

```
qaagent -si -c "DBF=qaclient.db"
```

## **-su option**

Function

Upgrades a client message store from SQL Anywhere Studio version 9.0.1 to version 9.0.2.

Syntax

```
qaagent -su -c “connection-string” ...
```

Remarks

This option is useful when you have a client message store with messages in it that was created with SQL Anywhere Studio 9.0.1.

Specify the database to upgrade in the connection string. Exits when done. This operation cannot be undone.

## **-v option**

Function

Allows you to specify what information is logged to the message log file and displayed in the synchronization window. A high level of verbosity may affect performance and should normally be used in the development phase only.

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax   | <b>qaagent -v levels ...</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Default  | Minimal verbosity.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Remarks  | <p>The <code>-v</code> option affects the log files and synchronization window. You only have a message log if you specify <code>-o</code> or <code>-ot</code> on the <code>qaagent</code> command line.</p> <p>If you specify <code>-v</code> alone, a small amount of information is logged.</p> <p>The values of <i>levels</i> are as follows. You can use one or more of these options at once; for example, <code>-vlm</code>.</p> <ul style="list-style-type: none"> <li>◆ <b>+</b> Turn on all logging options.</li> <li>◆ <b>l</b> Show all MobiLink Listener logging. This causes the MobiLink Listener (dblsn) to start with verbosity level <code>-v3</code>.<br/> <span style="font-size: small;">☞ For more information, see the <code>-v</code> option in the “The Listener utility” [<i>MobiLink Server-Initiated Synchronization User’s Guide</i>, page 38].</span></li> <li>◆ <b>m</b> Show all dbmsync logging. This causes the Adaptive Server Anywhere synchronization client (dbmsync) to start with verbosity level <code>-v+</code>.<br/> <span style="font-size: small;">☞ For more information, see the dbmsync “<code>-v</code> option” [<i>MobiLink Clients</i>, page 150].</span></li> <li>◆ <b>n</b> Show all network status change notifications. the QAnywhere Agent receives these notifications from the Listener utility.</li> <li>◆ <b>p</b> Show all message push notifications. The QAnywhere Agent receives these notifications from the Listener utility via the QAnywhere server, which includes a MobiLink Notifier.</li> <li>◆ <b>q</b> Show the SQL that is used to represent the transmission rules.</li> <li>◆ <b>s</b> Show all the message synchronizations that are initialized by QAnywhere Agent.</li> </ul> |
| See also | <ul style="list-style-type: none"> <li>◆ “<code>-o</code> option” on page 86</li> <li>◆ “<code>-ot</code> option” on page 87</li> <li>◆ “<code>-on</code> option” on page 86</li> <li>◆ “<code>-os</code> option” on page 87</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## **-x option**

|          |                                                                                                               |
|----------|---------------------------------------------------------------------------------------------------------------|
| Function | Specify the network protocol and protocol options for communication with the MobiLink synchronization server. |
| Syntax   | <b>qaagent -x protocol [ ( protocol-options;... ) ...</b>                                                     |

---

*protocol:* **http**, **tcpip**, **https**, or **https\_fips**

*protocol-options:* *keyword=value*

Remarks

The `-x` option is required when the MobiLink synchronization server is not on the same device as the QAnywhere Agent.

You can specify `-x` multiple times. This allows you to set up failover to multiple MobiLink synchronization servers. When you set up failover, the QAnywhere Agent attempts the MobiLink synchronization servers in the order in which you enter them on the command line.

The QAnywhere Agent also has a Listener that receives notifications from the MobiLink synchronization server that messages are available at the server for synchronization. This Listener only uses the first MobiLink synchronization server that is specified, and does not fail over to others.

See also

- ◆ For a complete list of protocol options that you can set for communication with the MobiLink synchronization server, see the `dbmlsrv9` “`-x` option” [*MobiLink Administration Guide*, page 214].
- ◆ “[Encrypting the communication stream](#)” on page 98
- ◆ “[MobiLink Transport-Layer Security](#)” [*MobiLink Administration Guide*, page 165]
- ◆ “[Setting up a failover mechanism](#)” on page 53

---

## CHAPTER 6

# Writing Secure Messaging Applications

About this chapter      This chapter describes techniques for implementing a secure messaging solution.

|          |                                                             |             |
|----------|-------------------------------------------------------------|-------------|
| Contents | <b>Topic:</b>                                               | <b>page</b> |
|          | <a href="#">Creating a secure client message store</a>      | 96          |
|          | <a href="#">Encrypting the communication stream</a>         | 98          |
|          | <a href="#">Using password authentication with MobiLink</a> | 99          |

---

## Creating a secure client message store

To create a secure client message store, you can:

- ◆ Change the default passwords.
  - ☞ See “[Manage client message store passwords](#)” on page 96.
- ◆ Encrypt the contents of the message store.
  - ☞ See “[Encrypting the client message store](#)” on page 97.

### Example

First, create an Adaptive Server Anywhere database with an encryption key:

```
dbinit mystore.db -ek key
```

Next, initialize the database as a client message store:

```
qaagent -id mystore -si -c "dbf=mystore.db;dbkey=some_phrase"
```

Next, create a new remote user with DBA authority, and a password for this user. Revoke the default QAnywhere user and change the password of the default DBA user. Log in as user DBA with password SQL and execute the following SQL statements:

```
GRANT CONNECT TO secure_user IDENTIFIED BY secure_password
GRANT MEMBERSHIP IN GROUP ml_qa_user_group TO secure_user
GRANT REMOTE dba TO secure_user
REVOKE CONNECT FROM ml_qa_user
GRANT CONNECT TO dba IDENTIFIED BY new_dba_password
COMMIT
```

Next, start the QAnywhere Agent with the secure DBA user:

```
qaagent -id mystore -c "dbf=mystore.db;dbkey=some_
phrase;uid=secure_user;pwd=secure_password"
```

You also need to set the connection parameters for your remote applications. For example, the QAManager properties file should contain this line:

```
CONNECT_PARAMS=dbn=mystore;dbkey=some_phrase;uid=secure_
user;pwd=secure_password
```

## Manage client message store passwords

You should change the passwords for the default user IDs that were created for the message store. The default user ID DBA with password SQL is created for every Adaptive Server Anywhere database. In addition, the `qaagent -si` option creates a default user ID of `ml_qa_user`, and creates a default password of `qanywhere`. To change these passwords, use the GRANT statement.

☞ For more information, see “Changing a password” [*ASA Database Administration Guide*, page 434].

## Encrypting the client message store

The following command can be used to encrypt the client message store when you create it.

```
dbinit -ek encryption-key database-file
```

When a message store has been initialized with an encryption key, the encryption key is required to start the database server on the encrypted message store.

Use the following command to specify the encryption key to start the QAnywhere Agent with an encrypted message store. The QAnywhere Agent will automatically start the database server on the encrypted message store using the encryption key provided.

```
qaagent -c "DBF=database-file;DBKEY=encryption-key"
```

Any application can now access the encrypted message store through the QAnywhere client API. Note that, since the database server used to manage the message store is already running, the application does not need to provide the encryption key.

If the QAnywhere Agent is not running and an application needs to access an encrypted message store, the QAnywhere client API will automatically start the database server using the connection parameters specified in the QAnywhere Manager initialization file. In order to start the database server on an encrypted message store, the encryption key must be specified in the database connection parameters as follows.

```
CONNECT_PARAMS=DBF=database-file;DBKEY=encryption-key
```

---

# Encrypting the communication stream

The `qaagent -x` option can be used to specify a secure communication stream that the QAnywhere Agent can use to connect to a MobiLink synchronization server. It allows you to implement server authentication using server-side certificates, and it allows you to encrypt the communication stream using strong encryption.

☞ For more information, see “[-x option](#)” on page 93.

☞ You must set up transport-layer security for the MobiLink synchronization server as well. For more information, see “[MobiLink Transport-Layer Security](#)” [*MobiLink Administration Guide*, page 165].

### Separately licensable option required

Transport-layer security requires that you obtain the separately-licensable SQL Anywhere Studio security option and is subject to export regulations.

☞ To order this component, see “[Separately-licensable components](#)” [*Introducing SQL Anywhere Studio*, page 5].

## Examples

The following examples show how to establish a secure communication stream between the QAnywhere Agent and the MobiLink synchronization server. They use sample certificates that are installed when the SQL Anywhere Studio security option is installed.

### Secure TCP/IP using RSA:

```
dbmlsrv9 -x tcpip(security=rsa_
           tls(certificate=rsaserver.crt;certificate_
              password=test))
qaagent -x tcpip(security=rsa_tls(trusted_
           certificates=rsaroot.crt))
```

### Secure TCP/IP using ECC:

```
dbmlsrv9 -x tcpip(security=ecc_
           tls(certificate=sample.crt;certificate_
              password=tJl#m6+W))
qaagent -x tcpip(security=ecc_tls(trusted_
           certificates=eccroot.crt))
```

### Secure HTTP using HTTPS (only RSA certificates are supported for HTTPS):

```
dbmlsrv9 -x https(certificate=rsaserver.crt;certificate_
                 password=test)
qaagent -x https(trusted_certificates=rsaroot.crt)
```



## Using password authentication with MobiLink

Once you have established a secure communication stream between the remote device and the server, you may also want to authenticate the user of the device to ensure that they are allowed to communicate with the server.

☞ For more information, see:

- ◆ “-mp option” on page 85
- ◆ “Authenticating MobiLink Users” [*MobiLink Clients*, page 9]



---

## CHAPTER 7

# QAnywhere Transmission Rules

About this chapter

This chapter describes how to write transmission rules. You can create transmission rules on the server to define which messages should be downloaded to the client, and transmission rules on the client to define which messages should be uploaded to the server.

Contents

| <b>Topic:</b>                               | <b>page</b> |
|---------------------------------------------|-------------|
| <a href="#">Transmission rules</a>          | 102         |
| <a href="#">Schedule syntax</a>             | 105         |
| <a href="#">Transmission rule variables</a> | 110         |
| <a href="#">Delete rules</a>                | 118         |

---

## Transmission rules

Message transmission is the action of moving messages from a client message store to a server message store, or vice versa.

Message transmission is handled by the QAnywhere Agent and the MobiLink synchronization server:

- ◆ The QAnywhere Agent is connected to the client message store. It transmits messages to and from the MobiLink synchronization server.
- ◆ The MobiLink synchronization server is connected to the server message store. It receives message transmissions from QAnywhere Agents and transmits them to other QAnywhere Agents.

Message transmission can only take place between a client message store and a server message store. A message transmission can only occur when a QAnywhere Agent is connected to a MobiLink synchronization server.

Transmission rules allow you to specify when message transmission is to occur and which messages to transmit. You can also use them to specify when messages should be deleted from the message stores, if you do not want to use the default behavior.

You can specify transmission rules on the server and on the client. See:

- ◆ [“Client transmission rules” on page 102](#)
- ◆ [“Server transmission rules” on page 103](#)

The transmission rules file holds the following kinds of entry:

- ◆ **Rules** No more than one rule can be entered per line.  
Each rule must be entered on a single line, but you can use \ as a line continuation character.
- ◆ **Comments** Comments are indicated by a line beginning with either a # or ; character. Any characters on that line are ignored.

☞ For more information, see [“Schedule syntax” on page 105](#) and [“Condition syntax” on page 106](#).

You can also use transmission rules files to determine when messages are to be deleted from the message stores.

☞ For more information, see [“Delete rules” on page 118](#).

## Client transmission rules

Client transmission rules govern the behavior of messages going from the client to the server. Client transmission rules are handled by the QAnywhere Agent.

By default, QAnywhere messages are transmitted every 10 seconds. You can change and customize this behavior by specifying a transmission rules file as the transmission policy for the QAnywhere Agent.

The following partial qaagent command line shows how to specify a rules file for the QAnywhere Agent:

```
qaagent -policy myrules.txt ...
```

☞ For a complete description of how to write transmission rules, see [“Schedule syntax” on page 105](#).

☞ For more information about policies, see [“Determining when message transmission should occur on the client” on page 37](#).

#### Example

For example, the following client transmission rules file specifies that during business hours only small high priority messages should be sent, while outside of business hours, any message can be sent. This rule is automatic, which indicates that if the condition is satisfied, the message is transmitted immediately. This example demonstrates that conditions can use information derived from the message as well as other information such as the current time.

```
automatic = ( ias_ContentSize < 100000 and ias_Priority > 7 ) \
  or ias_CurrentDayOfWeek in ( 'Saturday', 'Sunday' ) \
  or ias_CurrentTime < '8:00AM' or ias_CurrentTime > '6:00PM'
```

## Server transmission rules

Server transmission rules govern the behavior of messages going from the server to the client. Server transmission rules are handled by the MobiLink synchronization server. They apply both when you are using push notifications and when you are not using notifications.

You can create a server transmission rules file and specify it with the `ianywhere.qa.server.transmissionRulesFile` property in your QAnywhere messaging properties file.

Server transmission rules must be specified for each client by preceding a section of rules with the client message store ID in square brackets.

☞ For more information about the message properties file, see [“-m option” \[MobiLink Administration Guide, page 201\]](#).

#### Example

Following is a sample server transmission rules file. In the following example, the rules apply only to the client identified by the client message store ID `sample_store_id`.

---

```

[sample_store_id]
; This rule governs when messages are transmitted to the client
; store with id sample_store_id.
;
;   ias_Priority >= 7
;
; Messages with priority 7 or greater should always be
; transmitted.
;
;   ias_ContentSize < 100
;
; Small messages, that is messages less than 100 characters or
; bytes in size, should always be transmitted.
;
;   ias_CurrentTime < '8:00am' or ias_CurrentTime > '6:00pm'
;
; Outside of business hours, messages should always be
; transmitted

auto = ias_Priority >= 7 or ias_ContentSize < 100 \
      or ias_CurrentTime < '8:00am' or ias_CurrentTime > '6:00pm'

[qanywhere]
; This rule governs when messages are transmitted to the client
; store with id qanywhere.
;
;   tm_Subject not like '%non-business%'
;
; Messages with the property tm_Subject set to a value that
; includes the phrase 'non-business should not be transmitted'
;
;   ias_CurrentTime < '8:00am' or ias_CurrentTime > '6:00pm'
;
; Outside of business hours, messages should always be
; transmitted

auto = tm_Subject not like '%non-business%' \
      or ias_CurrentTime < '8:00am' or ias_CurrentTime > '6:00pm'

```

## Schedule syntax

Schedules are used to specify times when conditions are to be evaluated. At those times, the corresponding condition is evaluated for all messages ready to be sent. Those messages satisfying the condition are sent at that time.

### Syntax

Each rule is of the following form:

```
schedules=condition
```

When the scheduled time occurs, the condition is applied to each message. If the message satisfies the condition, then the message is transmitted.

```
schedules : { AUTOMATIC | schedule-spec [, ... ] }
```

```
schedule-spec :
```

```
{ START TIME start-time | BETWEEN start-time AND end-time }
[ EVERY period { HOURS | MINUTES | SECONDS } ]
[ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
[ START DATE start-date ]
```

### Parameters

- ◆ **AUTOMATIC** **AUTOMATIC** indicates that conditions are evaluated whenever a message is available for transmitting. Messages that satisfy the corresponding condition are transmitted.
- ◆ **schedule-spec** Schedule specifications other than **AUTOMATIC** specify times when conditions are to be evaluated. At those scheduled times, the corresponding condition is evaluated for all messages ready to be transmitted. Those messages satisfying the condition are transmitted at that time.
- ◆ **START TIME** The first scheduled time for each day on which the event is scheduled. If a **START DATE** is specified, the **START TIME** refers to that date. If no **START DATE** is specified, the **START TIME** is on the current day (unless the time has passed) and each subsequent day (if the schedule includes **EVERY** or **ON**).
- ◆ **BETWEEN ... AND ...** A range of times during the day outside of which no scheduled times occur. If a **START DATE** is specified, the scheduled times do not occur until that date.
- ◆ **EVERY** An interval between successive scheduled events. Scheduled events occur only after the **START TIME** for the day, or in the range specified by **BETWEEN ... AND**.
- ◆ **ON** A list of days on which the scheduled events occur. The default is every day if **EVERY** is specified. Days can be specified as days of the week or days of the month.

---

Days of the week are Mon, Tues, and so on. You may also use the full forms of the day, such as Monday. You must use the full forms of the day names if the language you are using is not English, is not the language requested by the client in the connection string, and is not the language which appears in the server window.

Days of the month are integers from 0 to 31. A value of 0 represents the last day of any month.

- ◆ **START DATE** The date on which scheduled events are to start occurring. The default is the current date.

## Usage

You can create more than one schedule for a given condition. This permits complex schedules to be implemented.

A schedule specification is recurring if its definition includes EVERY or ON; if neither of these reserved words is used, the schedule specifies at most a single time. An attempt to create a non-recurring schedule for which the start time has passed generates an error.

Each time a scheduled time occurs, the associated condition is evaluated and then the next scheduled time and date is calculated.

The next scheduled time is computed by inspecting the schedule or schedules, and finding the next schedule time that is in the future. If a schedule specifies every minute, and it takes 65 seconds to evaluate the conditions, it runs every two minutes. If you want execution to overlap, you must create more than one rule.

1. If the EVERY clause is used, find whether the next scheduled time falls on the current day, and is before the end of the BETWEEN ... AND range. If so, that is the next scheduled time.
2. If the next scheduled time does not fall on the current day, find the next date on which the event is to be executed.
3. Find the START TIME for that date, or the beginning of the BETWEEN ... AND range.

The QAnywhere schedule syntax is derived from the Adaptive Server Anywhere CREATE EVENT schedule syntax.

Keywords are case insensitive.

## See also

- ◆ “CREATE EVENT statement” [*ASA SQL Reference*, page 351].

## Condition syntax

QAnywhere condition syntax uses a SQL-like syntax. An expression evaluates to true, false, or unknown. Messages are sent only if the condition



evaluates to true. If a condition is empty, all messages are judged to satisfy the condition.

Keywords and string comparisons are case insensitive.

#### Syntax

```
condition :
| expression operator expression
| arithmetic-expr [ NOT ] BETWEEN start-expr AND end-expr
| rule-variable [ NOT ] IN ( string-literal, ... )
| rule-variable [ NOT ] LIKE pattern [ ESCAPE escape-character ]
| rule-variable IS [ NOT ] NULL
```

#### Parameters

- ◆ **condition** An expression that evaluates to true, false, or unknown. Only messages for which the condition evaluates to true are judged to satisfy the condition.
- ◆ **expression** An arithmetic or conditional expression. Standard bracketing using parentheses indicates the order of evaluation within expressions.
  - Arithmetic expressions are composed of themselves, arithmetic operators, numeric identifiers, and numeric literals. Arithmetic operators, in order of precedence, are:
    - ◆ +, - (unary sign indicators)
    - ◆ \*, / (multiplication and division)
    - ◆ +, - (addition and subtraction)
  - Conditional expressions are composed of themselves, comparison operators, and logical operators. Comparison operators, in order of precedence, are:
    - ◆ = (equals)
    - ◆ > (greater than)
    - ◆ >= (greater than or equals)
    - ◆ < (less than)
    - ◆ <= (less than or equals)
    - ◆ <> (not equal)
 Logical operators, in order of precedence, are:
    - ◆ NOT
    - ◆ AND
    - ◆ OR
- ◆ **rule-variable** A QAnywhere rule-variable is a message header, a message property, or a client store property. The type of a property value in a condition corresponds to the type used to set the property. If a property that does not exist in a message is referenced, its value is NULL.
 

☞ For more information, see [“Transmission rule variables”](#) on page 110.

- 
- ◆ **string-literal** A string literal is a sequence of characters enclosed in single quotes, using a string encoding as specified by the QAnywhere Agent.

A single quote in a string literal is represented by doubled single quote, For example, the following are valid string literals:

```
'literal'
```

```
'literal''s'
```

- ◆ **numeric-literal** An exact numeric literal is a numeric value without a decimal point, such as 57, -957, and +62. The value range is  $-2^{63}$  to  $2^{63} - 1$ , or -9223372036854775808 to 9223372036854775807.

An approximate numeric literal is a numeric value in scientific notation, such as 7E3 and -57.9E2, or a numeric value with a decimal, such as 7., -95.7, and +6.2. The value range is 2.22507385850721e-308 to 1.79769313486231e+308.

- ◆ **boolean-literal** The boolean literals are TRUE and FALSE.

- ◆ **BETWEEN** The BETWEEN condition can evaluate as true, false, or unknown. Without the NOT keyword, the condition evaluates as TRUE if *arithmetic-expr* is greater than or equal to *start-expr* and less than or equal to *end-expr*.

The NOT keyword reverses the meaning of the condition but leaves UNKNOWN unchanged.

The BETWEEN condition is equivalent to a combination of two inequalities:

```
[ NOT ] ( arithmetic-expr >= start-expr AND arithmetic-expr <= end-expr )
```

For example:

- age BETWEEN 15 AND 19 is equivalent to age >=15 AND age <= 19
- age NOT BETWEEN 15 AND 19 is equivalent to age < 15 OR age > 19.
- ◆ **IN** The IN condition evaluates according to the following rules:
  - True if *rule-variable* is not null and equals at least one of the values.
  - Unknown if *rule-variable* is null and the values list is not empty, or if at least one of the values is null and expression does not equal any of the other values.
  - False if none of the values are null, and *rule-variable* does not equal any of the values.

The NOT keyword interchanges true and false.

For example:

- `Country IN ( 'UK', 'US', 'France' )` is true for 'UK' and false for 'Peru'. It is equivalent to the following:

```
( Country = 'UK' )      \
OR ( Country = 'US' )  \
OR ( Country = 'France' )
```

- `Country NOT IN ( 'UK', 'US', 'France' )` is false for 'UK' and true for 'Peru'. It is equivalent to the following:

```
NOT (      ( Country = 'UK' )      \
           OR ( Country = 'US' )    \
           OR ( Country = 'France' ) )
```

- If the rule-variable of an IN operation is NULL
- ◆ **LIKE** The LIKE condition can evaluate as true, false, or unknown. Without the NOT keyword, the condition evaluates as TRUE if *expression* matches the *pattern*. If either *expression* or *pattern* is NULL, this condition is unknown.

The NOT keyword reverses the meaning of the condition, but leaves UNKNOWN unchanged.

The *pattern* may contain any number of wildcards. The wildcards are:

| Wildcard       | Matches                               |
|----------------|---------------------------------------|
| _ (underscore) | Any one character                     |
| % (percent)    | Any string of zero or more characters |

For example:

- `phone LIKE 12%3` is true for '123' or '12993' and false for '1234'
- `word LIKE 's_d'` is true for 'sad' and false for 'said'
- `phone NOT LIKE '12%3'` is false for '123' or '12993' and true for '1234'
- ◆ **ESCAPE CHARACTER** A single character string literal whose character is used to escape the special meaning of the wildcard characters (`_`, `%`) in *pattern*. For example:
  - `underscored LIKE '\_%' ESCAPE '\'` is true for '`_myvar`' and false for '`myvar`'.
- ◆ **IS NULL** The IS NULL condition evaluates to true if the rule-variable is unknown; otherwise it evaluates to false. The NOT keyword reverses the meaning of the condition. This condition cannot evaluate to unknown.

---

## Transmission rule variables

QAnywhere transmission rule variables are used in condition syntax in transmission rules files. They can be used to define transmission rules and delete rules. There are three types of rule variable:

- ◆ Message headers
- ◆ Message properties
- ◆ Message store properties

### Message headers

The following message headers are pre-defined.

- ◆ **ias\_Address** The address of the message. For example, myclient\myqueue.
- ◆ **ias\_Originator** The client message store ID associated with the message sender.
- ◆ **ias\_Status** The status of the message. Values can be:
  - **ias\_ExpireState** The message expired before it could be received by the intended recipient.
  - **ias\_FinalState** The message is received or expired. Therefore,  $\geq$  **ias\_FinalState** means that the message is received or expired, and  $>$  **ias\_FinalState** means that the message is neither received nor expired.
  - **ias\_PendingState** The message has not yet been received by the intended recipient.
  - **ias\_Received** The message was received by the intended recipient.
- ◆ **ias\_StatusTime** The date and time when the message reached the current status.
- ◆ **ias\_Expires** The date and time when the message will expire if it is not delivered.
- ◆ **ias\_Priority** The priority of message: a number from 0 to 9.
- ◆ **ias\_ContentSize** The size of the message content. If the message is a text message, this is the number of characters. If the message is binary, this is the number of bytes.

## Message properties

QAnywhere allows you to define message properties using the C++ or .NET QAnywhere APIs. These properties are shared between applications connected to the same message store. They are also synchronized to the server message store so that they are available to the transmission rules used by a QAnywhere Agent connected to the same client message store. You define message properties in messages, and then reference them in transmission rules.

Message property names are case insensitive. You can use a sequence of letters and digits, but the first character must be a letter. The following names are reserved and may not be used as message property names:

- ◆ NULL
- ◆ TRUE
- ◆ FALSE
- ◆ NOT
- ◆ AND
- ◆ OR
- ◆ BETWEEN
- ◆ LIKE
- ◆ IN
- ◆ IS
- ◆ ESCAPE
- ◆ Any name beginning with **ias\_**

The following QAManager methods can be used to manage message properties.

C++ methods to manage  
message properties

---

```

qa_bool getBooleanProperty( qa_const_string name, qa_bool *
    value )
qa_bool setBooleanProperty( qa_const_string name, qa_bool value
    )
qa_bool getByteProperty( qa_const_string name, qa_byte * value )
qa_bool setByteProperty( qa_const_string name, qa_byte value )
qa_bool getShortProperty( qa_const_string name, qa_short * value
    )
qa_bool setShortProperty( qa_const_string name, qa_short value )
qa_bool getIntProperty( qa_const_string name, qa_int * value )
qa_bool setIntProperty( qa_const_string name, qa_int value )
qa_bool getLongProperty( qa_const_string name, qa_long * value )
qa_bool setLongProperty( qa_const_string name, qa_long value )
qa_bool getFloatProperty( qa_const_string name, qa_float * value
    )
qa_bool setFloatProperty( qa_const_string name, qa_float value )
qa_bool getDoubleProperty( qa_const_string name, qa_double *
    value )
qa_bool setDoubleProperty( qa_const_string name, qa_double value
    )
qa_int getStringProperty( qa_const_string name, qa_string value,
    qa_int len )
qa_bool setStringProperty( qa_const_string name, qa_const_string
    value )

```

☞ For more information, see [“QAnywhere C++ API Reference” on page 121](#).

## C# methods to manage message properties

```

Object GetProperty( String name )
void SetProperty( String name, Object value )
boolean GetBooleanProperty( String name )
void SetBooleanProperty( String name, boolean value )
byte GetByteProperty( String name )
void SetByteProperty( String name, byte value )
short GetShortProperty( String name )
void SetShortProperty( String name, short value )
int GetIntProperty( String name )
void SetIntProperty( String name, int value )
long GetLongProperty( String name )
void SetLongProperty( String name, long value )
float GetFloatProperty( String name )
void SetFloatProperty( String name, float value )
double GetDoubleProperty( String name )
void SetDoubleProperty( String name, double value )
String GetStringProperty( String name )
void SetStringProperty( String name, String value )

```

☞ For more information, see [“iAnywhere.QAnywhere.Client namespace” on page 173](#).

## Example

```

c++
QAManager *mgr = ...; // Init QAManager
QAMessage *msg = mgr->createTextMessage();
msg->setStringProperty( "tm_Subject", "Some message subject" );
mgr->putMessage( "myqueue", mgr );

c#
QAManager mgr = ...; // init QAManager
QAMessage msg = mgr.createTextMessage();
msg.setStringProperty( "tm_Subject", "Some message subject" );
mgr.putMessage( "myqueue", msg );

```

## Client store properties

There are two types of client store property:

- ◆ pre-defined
- ◆ user-defined

### Pre-defined client store properties

The following client store properties are pre-defined.

- ◆ **ias\_Network** Information about the current network in use. **ias\_Network** is a special property. It has a number of built-in attributes that provide information regarding the current network that is being used by the device. These attributes are automatically set by QAnywhere:
  - **ias\_Network.Adapter** The current name of the network card, if any. (The name of the network card that is assigned to the Adapter attribute is displayed in the Agent window when the network connection is established.)
  - **ias\_Network.RAS** The current RAS dial-up name, if any.
  - **ias\_Network.IP** The current IP address assigned to the device, if any.
  - **ias\_Network.MAC** The current MAC address of the network card being used, if any.
- ◆ **ias\_CurrentDayOfWeek** The current day of the week.
- ◆ **ias\_CurrentDayOfMonth** The current day of the month, from 1-31.
- ◆ **ias\_CurrentMonth** The current month, from 1-12.
- ◆ **ias\_CurrentYear** The current year.
- ◆ **ias\_CurrentDate** The current date.

A string can be compared against `ias_currentDate` if it is supplied in one of two ways:

- 
- as a string of format yyyy/mm/dd or yyyy-mm-dd, which is interpreted unambiguously.
  - as a string according to the DATE\_ORDER database option set for the client message store database.

☞ For more information, see “Setting options” [ASA Database Administration Guide, page 614] and “DATE\_ORDER option [compatibility]” [ASA Database Administration Guide, page 648].

- ◆ **ias\_CurrentTime** The current time.

A string can be compared against ias\_CurrentTime if the hours, minutes, and seconds are separated by colons in the format hh:mm:ss:sss. A 24-hour clock is assumed unless **am** or **pm** are specified.

## Custom client store properties

QAnywhere allows you to define your own client store properties using the C++ or .NET QAnywhere APIs. These properties are shared between applications connected to the same message store. They are also synchronized to the server message store so that they are available to the transmission rules used by a QAnywhere Agent connected to the same client message store.

Message store property names are case insensitive. You can use a sequence of letters and digits, but the first character must be a letter. The following names are reserved and may not be used as client store property names:

- ◆ NULL
- ◆ TRUE
- ◆ FALSE
- ◆ NOT
- ◆ AND
- ◆ OR
- ◆ BETWEEN
- ◆ LIKE
- ◆ IN
- ◆ IS
- ◆ ESCAPE
- ◆ Any name beginning with **ias\_**

The following QAManager methods can be used to manage client store properties.

C++ methods to manage  
client store properties



```

qa_bool getBooleanStoreProperty( qa_const_string name, qa_bool *
    value )
qa_bool setBooleanStoreProperty( qa_const_string name, qa_bool
    value )
qa_bool getByteStoreProperty( qa_const_string name, qa_byte *
    value )
qa_bool setByteStoreProperty( qa_const_string name, qa_byte
    value )
qa_bool getShortStoreProperty( qa_const_string name, qa_short *
    value )
qa_bool setShortStoreProperty( qa_const_string name, qa_short
    value )
qa_bool getIntStoreProperty( qa_const_string name, qa_int *
    value )
qa_bool setIntStoreProperty( qa_const_string name, qa_int value
    )
qa_bool getLongStoreProperty( qa_const_string name, qa_long *
    value )
qa_bool setLongStoreProperty( qa_const_string name, qa_long
    value )
qa_bool getFloatStoreProperty( qa_const_string name, qa_float *
    value )
qa_bool setFloatStoreProperty( qa_const_string name, qa_float
    value )
qa_bool getDoubleStoreProperty( qa_const_string name, qa_double
    * value )
qa_bool setDoubleStoreProperty( qa_const_string name, qa_double
    value )
qa_int getStringStoreProperty( qa_const_string name, qa_string
    value, qa_int len )
qa_bool setStringStoreProperty( qa_const_string name, qa_const_
    string value )

```

☞ For more information, see [“QAnywhere C++ API Reference” on page 121](#).

### C# methods to manage client store properties

```

Object GetStoreProperty( String name )
void SetStoreProperty( String name, Object value )
boolean GetBooleanStoreProperty( String name )
void SetBooleanStoreProperty( String name, boolean value )
byte GetByteStoreProperty( String name )
void SetByteStoreProperty( String name, byte value )
short GetShortStoreProperty( String name )
void SetShortStoreProperty( String name, short value )
int GetIntStoreProperty( String name )
void SetIntStoreProperty( String name, int value )
long GetLongStoreProperty( String name )
void SetLongStoreProperty( String name, long value )
float GetFloatStoreProperty( String name )
void SetFloatStoreProperty( String name, float value )
double GetDoubleStoreProperty( String name )
void SetDoubleStoreProperty( String name, double value )
String GetStringStoreProperty( String name )
void SetStringStoreProperty( String name, String value )

```

---

☞ For more information, see [“iAnywhere.QAnywhere.Client namespace” on page 173](#).

Client store properties can also have attributes. An attribute is specified by appending a dot after the property name followed by the attribute name. In the following example, the Object property has two attributes: Shape and Color. The value of the Shape attribute is Round and the value of the Color attribute is Blue.

```
C++
    setStoreStringProperty( "Object.Shape", "Round" );
    setStoreStringProperty( "Object.Colour", "Blue" );
C#
    SetStoreProperty( "Object.Shape", "Round" );
    SetStoreProperty( "Object.Color", "Blue" );
```

All client store properties have a Type attribute that initially has no value. The value of the Type attribute must be the name of another property. When setting the Type attribute of a property, the property inherits the attributes of the property being assigned to it. In the following example, the Object property inherits the attributes of the Circle property. Hence the value of Object.Shape is Round and the value of Object.Color is Blue.

```
C++
    setStoreStringProperty( "Circle.Shape", "Round" );
    setStoreStringProperty( "Circle.Color", "Blue" );
    setStoreStringProperty( "Object.Type", "Circle" );
C#
    SetStoreProperty( "Circle.Shape", "Round" );
    SetStoreProperty( "Circle.Color", "Blue" );
    SetStoreProperty( "Object.Type", "Circle" );
```

## Example

This section provides an example in C# of how you can use client store properties in transmission rules.

Assume you have a Windows laptop that has the following network connectivity options: LAN, Wireless LAN, and Wireless WAN. Access to the network via LAN is provided by a network card named “My LAN Card”. Access to the network via Wireless LAN is provided by a network card named “My Wireless LAN Card”. Access to the network via Wireless WAN is provided by a network card named “My Wireless WAN Card”.

*Note:* The names of network adapters are fixed when the card is plugged in and the driver is installed. To find the name of a particular network card, connect to the network through that adapter, and then run `qaagent` with the `-vn` option. The QAnywhere Agent will display the network adapter name as follows:

```
"Listener thread received message '[netstat]
network-adapter-name !...'
```

You want to develop a messaging application that sends all messages to the server when connected using LAN or Wireless LAN and only high priority messages when connected using Wireless WAN. You define high priority messages as those whose priority is greater than or equal to 7.

First, define three client store properties for each of the network types: LAN, WLAN, and WWAN. Each of these properties will be assigned a Cost attribute. The Cost attribute is a value between 1 and 3 and represents the cost incurred when using the network. A value of 1 represents the lowest cost.

```
QAManager    qa_manager;

qa_manager.SetStoreProperty( "LAN.Cost", "1" );
qa_manager.SetStoreProperty( "WLAN.Cost", "2" );
qa_manager.SetStoreProperty( "WWAN.Cost", "3" );
```

Next, define three client store properties, one for each network card that will be used. The property name must match the network card name. Assign the appropriate network classification to each property by assigning the network type to the Type attribute. Each property will therefore inherit the attributes of the network types assigned to them.

```
QAManager    qa_manager;

qa_manager.SetStoreProperty( "My LAN Card.Type", "LAN" );
qa_manager.SetStoreProperty( "My Wireless LAN Card.Type", "WLAN"
);
qa_manager.SetStoreProperty( "My Wireless WAN Card.Type", "WWAN"
);
```

When network connectivity is established, QAnywhere will automatically define the Adapter attribute of the `ias_Network` property to one of “My LAN Card”, “My Wireless LAN Card” or “My Wireless WAN Card”, depending on the network in use. Similarly, it will automatically set the Type attribute of the `ias_Network` property to one of “My LAN Card”, “My Wireless LAN Card” or “My Wireless WAN Card” so that the `ias_Network` property will inherit the attributes of the network being used.

Finally, create a transmission rules file with the following transmission rule.

```
ias_Network.Cost < 3 or ias_Priority >= 7
```

---

## Delete rules

Delete rules determine the persistence of messages in the client message store and the server message store. They are specified in transmission rules files.

### Client delete rules

By default, messages are deleted from the client message store when the final status of the message is determined to be received or expired. You may want messages to be deleted faster than that, or to hold on to messages after acknowledgement. You do that by creating a delete section in your client transmission rules file.

☞ For more information about client transmission rules, see [“Client transmission rules” on page 102](#).

Following is an example of the delete rules section in a client transmission rules file:

```
[system:delete]

; This rule governs when messages are deleted from the client
; store
;
;   start time '1:00am' on ( 'Sunday' )
;
; Messages are deleted every Sunday at 1:00AM.
;
;   ias_Status >= ias_FinalState
;
; Typically, messages are deleted when they reach a final
; state: received, unreceivable, expired, or cancelled.

start time '1:00am' on ( 'Sunday' ) = ias_Status >= ias_
    FinalState
```

☞ For an explanation of `ias_Status`, see [“Message headers” on page 110](#).

### Server delete rules

By default, messages are deleted from the server message store as soon as the message has been delivered and delivery is confirmed. You may want to keep messages longer for purposes such as auditing. You do that by creating a delete section in your server transmission rules file.

Server-side delete rules apply to all QAnywhere clients.

☞ For more information about server transmission rules, see [“Server transmission rules” on page 103](#).

Following is an example of the delete rules section in a server transmission rules file:

```
[system:delete]

; This rule governs when messages are deleted from the server
; store
;
;   start time '1:00am' on ( 'Sunday' )
;
; Messages are deleted every Sunday at 1:00AM.
;
;   ias_Status >= ias_FinalState
;
; Typically messages are deleted when they reach a final
; status: received, unreceivable, expired or cancelled.

start time '1:00am' on ( 'Sunday' ) = ias_Status >= ias_
    FinalState
```

☞ For an explanation of `ias_Status`, see [“Message headers” on page 110](#).



---

## CHAPTER 8

# QAnywhere C++ API Reference

About this chapter

This chapter describes the QAnywhere C++ API.

Contents

| <b>Topic:</b>                                | <b>page</b> |
|----------------------------------------------|-------------|
| <a href="#">Class AcknowledgementMode</a>    | 122         |
| <a href="#">Class MessageProperties</a>      | 123         |
| <a href="#">Class MessageType</a>            | 126         |
| <a href="#">Class QABinaryMessage</a>        | 127         |
| <a href="#">Class QAEError</a>               | 135         |
| <a href="#">Class QAManager</a>              | 138         |
| <a href="#">Class QAManagerBase</a>          | 141         |
| <a href="#">Class QAManagerFactory</a>       | 153         |
| <a href="#">Class QAMessage</a>              | 155         |
| <a href="#">Class QAMessageListener</a>      | 167         |
| <a href="#">Class QATextMessage</a>          | 168         |
| <a href="#">Class QATransactionalManager</a> | 171         |

---

## Class AcknowledgementMode

|          |                                                                                                                                                                                                                                                                                                                                 |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | public <b>AcknowledgementMode</b>                                                                                                                                                                                                                                                                                               |
| Remarks  | The acknowledgement modes supported by QAnywhere are transactional, implicit and explicit. The client application specifies the acknowledgement mode when creating its instance of the <a href="#">Class QAManager</a> .                                                                                                        |
| Members  | All members of AcknowledgementMode, including all inherited members. <ul style="list-style-type: none"><li>◆ <a href="#">“EXPLICIT_ACKNOWLEDGEMENT Variable” on page 122</a></li><li>◆ <a href="#">“IMPLICIT_ACKNOWLEDGEMENT Variable” on page 122</a></li><li>◆ <a href="#">“TRANSACTIONAL Variable” on page 122</a></li></ul> |

### EXPLICIT\_ACKNOWLEDGEMENT Variable

|          |                                                                                                                                  |
|----------|----------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | const qa_short <b>AcknowledgementMode::EXPLICIT_ACKNOWLEDGEMENT</b>                                                              |
| Remarks  | With explicit acknowledgement, messages are acknowledged by a call to one of the acknowledge methods of <code>QAManager</code> . |

### IMPLICIT\_ACKNOWLEDGEMENT Variable

|          |                                                                                                                  |
|----------|------------------------------------------------------------------------------------------------------------------|
| Synopsis | const qa_short <b>AcknowledgementMode::IMPLICIT_ACKNOWLEDGEMENT</b>                                              |
| Remarks  | With implicit acknowledgement, messages are acknowledged as soon as they are received by the client application. |

### TRANSACTIONAL Variable

|          |                                                                                                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | const qa_short <b>AcknowledgementMode::TRANSACTIONAL</b>                                                                                                                                             |
| Remarks  | This mode indicates that messages are only acknowledged as part of the ongoing transaction. Hence, only a call to the commit method of <code>QAManager</code> acknowledges all outstanding messages. |



## Class MessageProperties

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | <code>public <b>MessageProperties</b></code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Remarks  | This class defines constant values for useful message property names for sending messages to QAnywhere Server.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Members  | <p>All members of MessageProperties, including all inherited members.</p> <ul style="list-style-type: none"> <li>◆ <a href="#">“ABS_RETRY_TIMEOUT Variable” on page 123</a></li> <li>◆ <a href="#">“ADAPTER Variable” on page 123</a></li> <li>◆ <a href="#">“FROM_ADDR Variable” on page 123</a></li> <li>◆ <a href="#">“MSG_TYPE Variable” on page 123</a></li> <li>◆ <a href="#">“NETWORK Variable” on page 124</a></li> <li>◆ <a href="#">“NETWORK_STATUS Variable” on page 124</a></li> <li>◆ <a href="#">“RETRY_FAILED Variable” on page 124</a></li> <li>◆ <a href="#">“RETRY_FAILED_ADDR Variable” on page 124</a></li> <li>◆ <a href="#">“RETRY_FAILED_PRIORITY Variable” on page 124</a></li> <li>◆ <a href="#">“RETRY_MAX Variable” on page 124</a></li> <li>◆ <a href="#">“RETRY_TIMEOUT Variable” on page 125</a></li> </ul> |

### ABS\_RETRY\_TIMEOUT Variable

|          |                                                                                                                                                       |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | <code>const qa_string <b>MessageProperties::ABS_RETRY_TIMEOUT</b></code>                                                                              |
| Remarks  | Optional property for messages sent through a connector. The time at which send retries through the connector will be stopped and the send is failed. |

### ADAPTER Variable

|          |                                                                                                                        |
|----------|------------------------------------------------------------------------------------------------------------------------|
| Synopsis | <code>const qa_string <b>MessageProperties::ADAPTER</b></code>                                                         |
| Remarks  | For “system” queue messages, a delimited list of network adapters that can be used to connect to the QAnywhere server. |

### FROM\_ADDR Variable

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Synopsis | <code>const qa_string <b>MessageProperties::FROM_ADDR</b></code> |
| Remarks  | Optional property indicating the address of the sender.          |

### MSG\_TYPE Variable

|          |                                                                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | <code>const qa_string <b>MessageProperties::MSG_TYPE</b></code>                                                                                        |
| Remarks  | Indicates the type of the message. If a message does not have this property set, it is a regular data message (ie. <a href="#">REGULAR Variable</a> ). |

---

See Also [Class MessageType](#)

## NETWORK Variable

Synopsis `const qa_string MessageProperties::NETWORK`

Remarks For “system” queue messages, a delimited list of network names that can be used to connect to the QAnywhere server.

## NETWORK\_STATUS Variable

Synopsis `const qa_string MessageProperties::NETWORK_STATUS`

Remarks For “system” queue messages, the state of the network connection. Value is 1 if connected, 0 otherwise.

## RETRY\_FAILED Variable

Synopsis `const qa_string MessageProperties::RETRY_FAILED`

Remarks Set by the connector when sending a message to the RetryFailedAddress. The receiving client can use this property to identify messages for which re-sending failed.

## RETRY\_FAILED\_ADDR Variable

Synopsis `const qa_string MessageProperties::RETRY_FAILED_ADDR`

Remarks Optional property for messages sent through a connector. Once either the RetryMax or RetryTimeout is exceeded, if this property is set, the message will be sent to this address.

## RETRY\_FAILED\_PRIORITY Variable

Synopsis `const qa_string MessageProperties::RETRY_FAILED_PRIORITY`

Remarks Optional property for messages sent through a connector. If a message is sent to the RetryFailedAddress, the message priority will be set to this.

## RETRY\_MAX Variable

Synopsis `const qa_string MessageProperties::RETRY_MAX`

Remarks Optional property for messages sent through a connector. The maximum number of send retries at the connector before failing the send.

## RETRY\_TIMEOUT Variable

Synopsis

const qa\_string **MessageProperties::RETRY\_TIMEOUT**

Remarks

Optional property for messages sent through a connector. The duration after which send retries through the connector will be stopped and the send is failed.

---

## Class `MessageType`

|          |                                                                                                                                                                                                                                                                                                                             |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | <code>public <b>MessageType</b></code>                                                                                                                                                                                                                                                                                      |
| Remarks  | This class defines constant values for the message property “ias_MessageType”.                                                                                                                                                                                                                                              |
| See Also | <a href="#">MSG_TYPE Variable</a>                                                                                                                                                                                                                                                                                           |
| Members  | All members of <code>MessageType</code> , including all inherited members. <ul style="list-style-type: none"><li>◆ <a href="#">“NETWORK_STATUS_NOTIFICATION Variable” on page 126</a></li><li>◆ <a href="#">“PUSH_NOTIFICATION Variable” on page 126</a></li><li>◆ <a href="#">“REGULAR Variable” on page 126</a></li></ul> |

### `NETWORK_STATUS_NOTIFICATION` Variable

|          |                                                                                     |
|----------|-------------------------------------------------------------------------------------|
| Synopsis | <code>const qa_int <b>MessageType::NETWORK_STATUS_NOTIFICATION</b></code>           |
| Remarks  | Message is a network status notification. Indicates a change in the network status. |

### `PUSH_NOTIFICATION` Variable

|          |                                                                                                           |
|----------|-----------------------------------------------------------------------------------------------------------|
| Synopsis | <code>const qa_int <b>MessageType::PUSH_NOTIFICATION</b></code>                                           |
| Remarks  | Message is a push notification. Indicates that a message is waiting to be sent from the QAnywhere server. |

### `REGULAR` Variable

|          |                                                       |
|----------|-------------------------------------------------------|
| Synopsis | <code>const qa_int <b>MessageType::REGULAR</b></code> |
| Remarks  | Regular data message.                                 |

# Class QABinaryMessage

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | public <b>QABinaryMessage</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Base classes | ◆ “Class QAMessage” on page 155                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Remarks      | <p>An QABinaryMessage object is used to send a message containing a stream of uninterpreted bytes. It inherits from the <a href="#">Class QAMessage</a> class and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes.</p> <p>When the message is first created, the body of the message is in write-only mode. After the first call to reset has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called reset so that the message body is in read-only mode for the client.</p> <p>If a client attempts to write a message in read-only mode, a <code>COMMON_MSG_NOT_WRITEABLE_ERROR</code> is set.</p>                                                                                                                                                                                                                                                                                                                                                                 |
| Members      | <p>All members of QABinaryMessage, including all inherited members.</p> <ul style="list-style-type: none"> <li>◆ “castToBinaryMessage Function” on page 156</li> <li>◆ “castToTextMessage Function” on page 156</li> <li>◆ “clearProperties Function” on page 156</li> <li>◆ “DEFAULT_PRIORITY Variable” on page 156</li> <li>◆ “DEFAULT_TIME_TO_LIVE Variable” on page 156</li> <li>◆ “getAddress Function” on page 157</li> <li>◆ “getBodyLength Function” on page 129</li> <li>◆ “getBooleanProperty Function” on page 157</li> <li>◆ “getByteProperty Function” on page 157</li> <li>◆ “getDoubleProperty Function” on page 157</li> <li>◆ “getExpiration Function” on page 158</li> <li>◆ “getFloatProperty Function” on page 158</li> <li>◆ “getInReplyToID Function” on page 158</li> <li>◆ “getIntProperty Function” on page 158</li> <li>◆ “getLongProperty Function” on page 159</li> <li>◆ “getMessageID Function” on page 159</li> <li>◆ “getPriority Function” on page 159</li> <li>◆ “getPropertyNames Function” on page 159</li> <li>◆ “getPropertyType Function” on page 160</li> <li>◆ “getRedelivered Function” on page 160</li> <li>◆ “getReplyToAddress Function” on page 160</li> <li>◆ “getShortProperty Function” on page 160</li> </ul> |

- 
- ◆ “getStringProperty Function” on page 161
  - ◆ “getStringProperty Function” on page 161
  - ◆ “getTimestamp Function” on page 161
  - ◆ “getTimestampAsString Function” on page 162
  - ◆ “propertyExists Function” on page 162
  - ◆ “readBinary Function” on page 129
  - ◆ “readBoolean Function” on page 129
  - ◆ “readByte Function” on page 129
  - ◆ “readChar Function” on page 130
  - ◆ “readDouble Function” on page 130
  - ◆ “readFloat Function” on page 130
  - ◆ “readInt Function” on page 130
  - ◆ “readLong Function” on page 131
  - ◆ “readShort Function” on page 131
  - ◆ “readString Function” on page 131
  - ◆ “reset Function” on page 131
  - ◆ “setAddress Function” on page 162
  - ◆ “setBooleanProperty Function” on page 163
  - ◆ “setByteProperty Function” on page 163
  - ◆ “setDoubleProperty Function” on page 163
  - ◆ “setFloatProperty Function” on page 163
  - ◆ “setInReplyToID Function” on page 164
  - ◆ “setIntProperty Function” on page 164
  - ◆ “setLongProperty Function” on page 164
  - ◆ “setMessageID Function” on page 164
  - ◆ “setPriority Function” on page 165
  - ◆ “setRedelivered Function” on page 165
  - ◆ “setReplyToAddress Function” on page 165
  - ◆ “setShortProperty Function” on page 165
  - ◆ “setStringProperty Function” on page 166
  - ◆ “setTimestamp Function” on page 166
  - ◆ “writeBinary Function” on page 132
  - ◆ “writeBoolean Function” on page 132
  - ◆ “writeByte Function” on page 132
  - ◆ “writeChar Function” on page 132
  - ◆ “writeDouble Function” on page 133
  - ◆ “writeFloat Function” on page 133
  - ◆ “writeInt Function” on page 133
  - ◆ “writeLong Function” on page 133
  - ◆ “writeShort Function” on page 133
  - ◆ “writeString Function” on page 134
  - ◆ “~QABinaryMessage Function” on page 134
  - ◆ “~QAMessage Function” on page 166

## getBodyLength Function

Synopsis                      virtual qa\_long **QABinaryMessage::getBodyLength()**

Remarks                     Returns the size in qa\_bytes of the message body.

## readBinary Function

Synopsis                      virtual qa\_int **QABinaryMessage::readBinary**(  
                                  qa\_bytes *value*  
                                  qa\_int *length*  
                                  )

Parameters                  ♦ **value**    the buffer into which the data is read  
                                  ♦ **length**    the maximum number of bytes to read

Remarks                     Reads a portion of the bytes message stream.

Returns                       the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

## readBoolean Function

Synopsis                      virtual qa\_bool **QABinaryMessage::readBoolean**(  
                                  qa\_bool \* *value*  
                                  )

Parameters                  ♦ **value**    the destination of the qa\_bool value read from the bytes message stream

Remarks                     Reads a qa\_bool from the bytes message stream.

Returns                       true if and only if the operation succeeded

## readByte Function

Synopsis                      virtual qa\_bool **QABinaryMessage::readByte**(  
                                  qa\_byte \* *value*  
                                  )

Parameters                  ♦ **value**    the destination of the qa\_byte value read from the bytes message stream

Remarks                     Reads a signed 8-bit value from the bytes message stream.

Returns                       true if and only if the operation succeeded

---

## readChar Function

|            |                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QABinaryMessage::readChar</b>(     qa_char * <i>value</i> )</pre>           |
| Parameters | ◆ <b>value</b> the destination of the <code>qa_char</code> value read from the bytes message stream |
| Remarks    | Reads a character value from the bytes message stream.                                              |
| Returns    | true if and only if the operation succeeded                                                         |

## readDouble Function

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QABinaryMessage::readDouble</b>(     qa_double * <i>value</i> )</pre>         |
| Parameters | ◆ <b>value</b> the destination of the <code>qa_double</code> value read from the bytes message stream |
| Remarks    | Reads a double from the bytes message stream.                                                         |
| Returns    | true if and only if the operation succeeded                                                           |

## readFloat Function

|            |                                                                                                      |
|------------|------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QABinaryMessage::readFloat</b>(     qa_float * <i>value</i> )</pre>          |
| Parameters | ◆ <b>value</b> the destination of the <code>qa_float</code> value read from the bytes message stream |
| Remarks    | Reads a float from the bytes message stream.                                                         |
| Returns    | true if and only if the operation succeeded                                                          |

## readInt Function

|            |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QABinaryMessage::readInt</b>(     qa_int * <i>value</i> )</pre>            |
| Parameters | ◆ <b>value</b> the destination of the <code>qa_int</code> value read from the bytes message stream |
| Remarks    | Reads a signed 32-bit integer from the bytes message stream.                                       |
| Returns    | true if and only if the operation succeeded                                                        |



## readLong Function

|            |                                                                                        |
|------------|----------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QBinaryMessage::readLong</b> (<br>qa_long * <i>value</i><br>)       |
| Parameters | ◆ <b>value</b> the destination of the qa_long value read from the bytes message stream |
| Remarks    | Reads a signed 64-bit integer from the bytes message stream.                           |
| Returns    | true if and only if the operation succeeded                                            |

## readShort Function

|            |                                                                                         |
|------------|-----------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QBinaryMessage::readShort</b> (<br>qa_short * <i>value</i><br>)      |
| Parameters | ◆ <b>value</b> the destination of the qa_short value read from the bytes message stream |
| Remarks    | Reads a signed 16-bit number from the bytes message stream.                             |
| Returns    | true if and only if the operation succeeded                                             |

## readString Function

|            |                                                                                                                                                                                              |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_int <b>QBinaryMessage::readString</b> (<br>qa_string <i>dest</i><br>qa_int <i>maxLen</i><br>)                                                                                     |
| Parameters | ◆ <b>dest</b> the destination of the qa_string value read from the bytes message stream<br><br>◆ <b>maxLen</b> the maximum number of qa_chars to read, including the null terminator qa_char |
| Remarks    | Reads a string from the bytes message stream.                                                                                                                                                |
| Returns    | the total number of non-null qa_chars read into the buffer, or -1 if there is no more data, or the buffer is too small                                                                       |

## reset Function

|          |                                                                                               |
|----------|-----------------------------------------------------------------------------------------------|
| Synopsis | virtual void <b>QBinaryMessage::reset</b> ()                                                  |
| Remarks  | Puts the message body in read-only mode and repositions the stream of bytes to the beginning. |

---

## writeBinary Function

|            |                                                                                                                                                                                                                         |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual void <b>QABinaryMessage::writeBinary</b>(     qa_const_bytes <i>value</i>     qa_int <i>offset</i>     qa_int <i>length</i> )</pre>                                                                        |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>value</b> the byte array value to be written</li><li>◆ <b>offset</b> the initial offset within the byte array</li><li>◆ <b>length</b> the number of bytes to write</li></ul> |
| Remarks    | Writes a portion of a byte array to the bytes message stream.                                                                                                                                                           |

## writeBoolean Function

|            |                                                                                                                                                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual void <b>QABinaryMessage::writeBoolean</b>(     qa_bool <i>value</i> )</pre>                                                                                                                                                  |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>value</b> the <code>qa_bool</code> value to be written</li></ul>                                                                                                                               |
| Remarks    | Writes a <code>qa_bool</code> to the bytes message stream as a 1-byte value. The value <code>true</code> is written as the value <code>(qa_byte)1</code> ; the value <code>false</code> is written as the value <code>(qa_byte)0</code> . |

## writeByte Function

|            |                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual void <b>QABinaryMessage::writeByte</b>(     qa_byte <i>value</i> )</pre>                       |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>value</b> the <code>qa_byte</code> value to be written</li></ul> |
| Remarks    | Writes a <code>qa_byte</code> to the bytes message stream as a 1-byte value.                                |

## writeChar Function

|            |                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual void <b>QABinaryMessage::writeChar</b>(     qa_char <i>value</i> )</pre>                       |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>value</b> the <code>qa_char</code> value to be written</li></ul> |
| Remarks    | Writes a <code>qa_char</code> to the bytes message stream as a 2-byte value, high byte first.               |

## writeDouble Function

|            |                                                                                                                                                     |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QABinaryMessage::writeDouble</b> (<br>qa_double <i>value</i><br>)                                                                   |
| Parameters | ◆ <b>value</b> the qa_double to be written                                                                                                          |
| Remarks    | Converts the qa_double argument to a qa_long and then writes that qa_long value to the bytes message stream as an 8-byte quantity, high byte first. |

## writeFloat Function

|            |                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QABinaryMessage::writeFloat</b> (<br>qa_float <i>value</i><br>)                                                                 |
| Parameters | ◆ <b>value</b> the qa_float to be written                                                                                                       |
| Remarks    | Converts the qa_float argument to a qa_int and then writes that qa_int value to the bytes message stream as a 4-byte quantity, high byte first. |

## writeInt Function

|            |                                                                             |
|------------|-----------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QABinaryMessage::writeInt</b> (<br>qa_int <i>value</i><br>) |
| Parameters | ◆ <b>value</b> the qa_int to be written                                     |
| Remarks    | Writes a qa_int to the bytes message stream as four bytes, high byte first. |

## writeLong Function

|            |                                                                               |
|------------|-------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QABinaryMessage::writeLong</b> (<br>qa_long <i>value</i><br>) |
| Parameters | ◆ <b>value</b> the qa_long to be written                                      |
| Remarks    | Writes a qa_long to the bytes message stream as eight bytes, high byte first. |

## writeShort Function

|            |                                                                                 |
|------------|---------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QABinaryMessage::writeShort</b> (<br>qa_short <i>value</i><br>) |
| Parameters | ◆ <b>value</b> the qa_short to be written                                       |

---

Remarks                      Writes a `qa_short` to the bytes message stream as two bytes, high byte first.

## writeString Function

Synopsis                      virtual void **QABinaryMessage::writeString**(  
                                 `qa_const_string value`  
                                 )

Parameters                    ♦ **value**    the string to be written

Remarks                      Writes a string to the bytes message stream.

## ~QABinaryMessage Function

Synopsis                      virtual **QABinaryMessage::~QABinaryMessage**()

Remarks                      Virtual destructor.

## Class QAEError

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | public <b>QAEError</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Remarks  | This class defines error constants associated with the QAnywhere client. A QAEError object is used internally by the <a href="#">Class QAManager</a> object to keep track of errors associated with messaging operations. The application programmer should not need to create an instance of this class. The error constants should be used by the application programmer when interpreting error codes returned by <a href="#">getLastError Function</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| See Also | <a href="#">getLastErrorMsg Function</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Members  | All members of QAEError, including all inherited members. <ul style="list-style-type: none"> <li>◆ “COMMON_GET_INIT_FILE_ERROR Variable” on page 135</li> <li>◆ “COMMON_INIT_ERROR Variable” on page 135</li> <li>◆ “COMMON_INIT_THREAD_ERROR Variable” on page 136</li> <li>◆ “COMMON_INVALID_PROPERTY Variable” on page 136</li> <li>◆ “COMMON_MSG_NOT_WRITEABLE_ERROR Variable” on page 136</li> <li>◆ “COMMON_MSG_RETRIEVE_ERROR Variable” on page 136</li> <li>◆ “COMMON_MSG_STORE_ERROR Variable” on page 136</li> <li>◆ “COMMON_MSG_STORE_NOT_INITIALIZED Variable” on page 136</li> <li>◆ “COMMON_MSG_STORE_TOO_LARGE Variable” on page 136</li> <li>◆ “COMMON_NO_DEST_ERROR Variable” on page 136</li> <li>◆ “COMMON_NO_IMPLEMENTATION Variable” on page 137</li> <li>◆ “COMMON_OPEN_ERROR Variable” on page 137</li> <li>◆ “COMMON_OPEN_LOG_FILE_ERROR Variable” on page 137</li> <li>◆ “COMMON_TERMINATE_ERROR Variable” on page 137</li> <li>◆ “COMMON_UNEXPECTED_EOM_ERROR Variable” on page 137</li> <li>◆ “QA_NO_ERROR Variable” on page 137</li> <li>◆ “~QAEError Function” on page 137</li> </ul> |

### COMMON\_GET\_INIT\_FILE\_ERROR Variable

|          |                                                          |
|----------|----------------------------------------------------------|
| Synopsis | const qa_int <b>QAEError::COMMON_GET_INIT_FILE_ERROR</b> |
| Remarks  | Unable to access client properties file.                 |

### COMMON\_INIT\_ERROR Variable

|          |                                                 |
|----------|-------------------------------------------------|
| Synopsis | const qa_int <b>QAEError::COMMON_INIT_ERROR</b> |
| Remarks  | Initialization error.                           |

---

## **COMMON\_INIT\_THREAD\_ERROR Variable**

Synopsis                    `const qa_int QLError::COMMON_INIT_THREAD_ERROR`

Remarks                  Error initializing background thread.

## **COMMON\_INVALID\_PROPERTY Variable**

Synopsis                    `const qa_int QLError::COMMON_INVALID_PROPERTY`

Remarks                  There is an invalid property in the client properties file.

## **COMMON\_MSG\_NOT\_WRITEABLE\_ERROR Variable**

Synopsis                    `const qa_int QLError::COMMON_MSG_NOT_WRITEABLE_ERROR`

Remarks                  Message is not writeable.

## **COMMON\_MSG\_RETRIEVE\_ERROR Variable**

Synopsis                    `const qa_int QLError::COMMON_MSG_RETRIEVE_ERROR`

Remarks                  Error retrieving message from message store.

## **COMMON\_MSG\_STORE\_ERROR Variable**

Synopsis                    `const qa_int QLError::COMMON_MSG_STORE_ERROR`

Remarks                  Error storing message to message store.

## **COMMON\_MSG\_STORE\_NOT\_INITIALIZED Variable**

Synopsis                    `const qa_int QLError::COMMON_MSG_STORE_NOT_INITIALIZED`

Remarks                  The message store has not been initialized for messaging.

## **COMMON\_MSG\_STORE\_TOO\_LARGE Variable**

Synopsis                    `const qa_int QLError::COMMON_MSG_STORE_TOO_LARGE`

Remarks                  The message store is too large relative to the disk free space on the device.

## **COMMON\_NO\_DEST\_ERROR Variable**

Synopsis                    `const qa_int QLError::COMMON_NO_DEST_ERROR`

Remarks No destination.

### **COMMON\_NO\_IMPLEMENTATION Variable**

Synopsis `const qa_int QAEError::COMMON_NO_IMPLEMENTATION`

Remarks The function is not implemented.

### **COMMON\_OPEN\_ERROR Variable**

Synopsis `const qa_int QAEError::COMMON_OPEN_ERROR`

Remarks Error opening connection to message store.

### **COMMON\_OPEN\_LOG\_FILE\_ERROR Variable**

Synopsis `const qa_int QAEError::COMMON_OPEN_LOG_FILE_ERROR`

Remarks Error opening the log file.

### **COMMON\_TERMINATE\_ERROR Variable**

Synopsis `const qa_int QAEError::COMMON_TERMINATE_ERROR`

Remarks Termination error.

### **COMMON\_UNEXPECTED\_EOM\_ERROR Variable**

Synopsis `const qa_int QAEError::COMMON_UNEXPECTED_EOM_ERROR`

Remarks Unexpected end of message reached.

### **QA\_NO\_ERROR Variable**

Synopsis `const qa_int QAEError::QA_NO_ERROR`

Remarks No error.

### **~QAEError Function**

Synopsis `virtual QAEError::~~QAEError()`

Remarks Virtual destructor.

---

# Class QAManager

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | public <b>QAManager</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Base classes | ◆ “Class QAManagerBase” on page 141                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Remarks      | This class is the manager for non-transactional messaging.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Members      | All members of QAManager, including all inherited members. <ul style="list-style-type: none"><li>◆ “acknowledge Function” on page 139</li><li>◆ “acknowledgeAll Function” on page 139</li><li>◆ “acknowledgeUntil Function” on page 139</li><li>◆ “close Function” on page 142</li><li>◆ “createBinaryMessage Function” on page 142</li><li>◆ “createTextMessage Function” on page 142</li><li>◆ “deleteMessage Function” on page 142</li><li>◆ “getBooleanStoreProperty Function” on page 143</li><li>◆ “getByteStoreProperty Function” on page 143</li><li>◆ “getDoubleStoreProperty Function” on page 143</li><li>◆ “getFloatStoreProperty Function” on page 144</li><li>◆ “getIntStoreProperty Function” on page 144</li><li>◆ “getLastError Function” on page 144</li><li>◆ “getLastErrorMsg Function” on page 144</li><li>◆ “getLongStoreProperty Function” on page 145</li><li>◆ “getMessage Function” on page 145</li><li>◆ “getMessageNoWait Function” on page 145</li><li>◆ “getMessageTimeout Function” on page 145</li><li>◆ “getMode Function” on page 146</li><li>◆ “getShortStoreProperty Function” on page 146</li><li>◆ “getStringStoreProperty Function” on page 146</li><li>◆ “open Function” on page 139</li><li>◆ “peekFirstMessage Function” on page 147</li><li>◆ “peekNextMessage Function” on page 147</li><li>◆ “publishMessage Function” on page 147</li><li>◆ “putMessage Function” on page 147</li><li>◆ “putMessageTimeToLive Function” on page 148</li><li>◆ “recover Function” on page 140</li><li>◆ “setBooleanStoreProperty Function” on page 148</li><li>◆ “setByteStoreProperty Function” on page 148</li><li>◆ “setDoubleStoreProperty Function” on page 149</li><li>◆ “setFloatStoreProperty Function” on page 149</li><li>◆ “setIntStoreProperty Function” on page 149</li><li>◆ “setLongStoreProperty Function” on page 150</li><li>◆ “setMessageListener Function” on page 150</li></ul> |



- ◆ “setProperty Function” on page 150
- ◆ “setShortStoreProperty Function” on page 151
- ◆ “setStringStoreProperty Function” on page 151
- ◆ “start Function” on page 151
- ◆ “stop Function” on page 151
- ◆ “triggerSendReceive Function” on page 151
- ◆ “QAManager Function” on page 140
- ◆ “QAManagerBase Function” on page 152

## acknowledge Function

|            |                                                                                |
|------------|--------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManager::acknowledge</b> (<br>QAMessage * <i>msg</i><br>) |
| Parameters | ◆ <b>msg</b> the message                                                       |
| Remarks    | Acknowledges the given message.                                                |
| Returns    | true if and only if the operation succeeded                                    |

## acknowledgeAll Function

|          |                                                     |
|----------|-----------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAManager::acknowledgeAll</b> () |
| Remarks  | Acknowledges all messages.                          |
| Returns  | true if and only if the operation succeeded         |

## acknowledgeUntil Function

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManager::acknowledgeUntil</b> (<br>QAMessage * <i>msg</i><br>) |
| Parameters | ◆ <b>msg</b> the message                                                            |
| Remarks    | Acknowledges the given message and all previous messages.                           |
| Returns    | true if and only if the operation succeeded                                         |

## open Function

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManager::open</b> (<br>qa_short <i>mode</i><br>) |
| Parameters | ◆ <b>mode</b> the acknowledge mode                                    |

---

Remarks                    Opens the QAManager with the given acknowledge mode.  
Returns                    true if and only if the operation succeeded  
See Also                    [Class AcknowledgementMode](#)

## recover Function

Synopsis                    virtual qa\_bool **QAManager::recover()**  
Remarks                    Recovers all unacknowledged messages.  
Returns                    true if and only if the operation succeeded

## ~QAManager Function

Synopsis                    virtual **QAManager::~~QAManager()**  
Remarks                    Virtual destructor.

# Class QAManagerBase

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis        | public <b>QAManagerBase</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Derived classes | <ul style="list-style-type: none"> <li>◆ “Class <a href="#">QAManager</a>” on page 138</li> <li>◆ “Class <a href="#">QATransactionalManager</a>” on page 171</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Remarks         | <p>This class acts as a base class for <a href="#">Class QATransactionalManager</a> and <a href="#">Class QAManager</a>, which manage transactional and non-transactional messaging respectively. There must be a single instance of QAManagerBase per thread in the application. This class is also a factory for creating messages. Since the <a href="#">getMessage Function</a> methods also create messages, this class manages all messages, so that they can be released from memory either at a user method call, or when the QAManagerBase is closed. The <a href="#">publishMessage Function</a> methods will always return false, since the publish/subscribe model is not supported.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Members         | <p>All members of QAManagerBase, including all inherited members.</p> <ul style="list-style-type: none"> <li>◆ “<a href="#">close Function</a>” on page 142</li> <li>◆ “<a href="#">createBinaryMessage Function</a>” on page 142</li> <li>◆ “<a href="#">createTextMessage Function</a>” on page 142</li> <li>◆ “<a href="#">deleteMessage Function</a>” on page 142</li> <li>◆ “<a href="#">getBooleanStoreProperty Function</a>” on page 143</li> <li>◆ “<a href="#">getByteStoreProperty Function</a>” on page 143</li> <li>◆ “<a href="#">getDoubleStoreProperty Function</a>” on page 143</li> <li>◆ “<a href="#">getFloatStoreProperty Function</a>” on page 144</li> <li>◆ “<a href="#">getIntStoreProperty Function</a>” on page 144</li> <li>◆ “<a href="#">getLastError Function</a>” on page 144</li> <li>◆ “<a href="#">getLastErrorMsg Function</a>” on page 144</li> <li>◆ “<a href="#">getLongStoreProperty Function</a>” on page 145</li> <li>◆ “<a href="#">getMessage Function</a>” on page 145</li> <li>◆ “<a href="#">getMessageNoWait Function</a>” on page 145</li> <li>◆ “<a href="#">getMessageTimeout Function</a>” on page 145</li> <li>◆ “<a href="#">getMode Function</a>” on page 146</li> <li>◆ “<a href="#">getShortStoreProperty Function</a>” on page 146</li> <li>◆ “<a href="#">getStringStoreProperty Function</a>” on page 146</li> <li>◆ “<a href="#">peekFirstMessage Function</a>” on page 147</li> <li>◆ “<a href="#">peekNextMessage Function</a>” on page 147</li> <li>◆ “<a href="#">publishMessage Function</a>” on page 147</li> <li>◆ “<a href="#">putMessage Function</a>” on page 147</li> <li>◆ “<a href="#">putMessageTimeToLive Function</a>” on page 148</li> <li>◆ “<a href="#">setBooleanStoreProperty Function</a>” on page 148</li> <li>◆ “<a href="#">setByteStoreProperty Function</a>” on page 148</li> <li>◆ “<a href="#">setDoubleStoreProperty Function</a>” on page 149</li> </ul> |

- ◆ “setFloatStoreProperty Function” on page 149
- ◆ “setIntStoreProperty Function” on page 149
- ◆ “setLongStoreProperty Function” on page 150
- ◆ “setMessageListener Function” on page 150
- ◆ “setProperty Function” on page 150
- ◆ “setShortStoreProperty Function” on page 151
- ◆ “setStringStoreProperty Function” on page 151
- ◆ “start Function” on page 151
- ◆ “stop Function” on page 151
- ◆ “triggerSendReceive Function” on page 151
- ◆ “QAManagerBase Function” on page 152

## close Function

|          |                                                                                                                                                                                                                  |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAManagerBase::close()</b>                                                                                                                                                                    |
| Remarks  | Closes the QAManagerBase. This releases all resources associated with the instance. When an instance of QAManagerBase is closed, it cannot be re-opened; a new instance must be created and opened in this case. |
| Returns  | true if and only if the operation succeeded                                                                                                                                                                      |

## createBinaryMessage Function

|          |                                                                                                                                  |
|----------|----------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual QABinaryMessage * <b>QAManagerBase::createBinaryMessage()</b>                                                            |
| Remarks  | Creates a QABinaryMessage object. A QABinaryMessage object is used to send a message containing a stream of uninterpreted bytes. |
| Returns  | the message that was created                                                                                                     |

## createTextMessage Function

|          |                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual QATextMessage * <b>QAManagerBase::createTextMessage()</b>                                              |
| Remarks  | Creates a QATextMessage object. A QATextMessage object is used to send a message containing a qa_string value. |
| Returns  | the message that was created                                                                                   |

## deleteMessage Function

|            |                                                                        |
|------------|------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAManagerBase::deleteMessage(QAMessage * msg)</b><br>) |
| Parameters | ◆ <b>msg</b> the message to be deleted                                 |

Remarks Deletes a `QAMessage` object. By default, messages created by the above methods are deleted automatically when the `QAManagerBase` is closed. This method allows more control over when messages are deleted.

## getBooleanStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::getBooleanStoreProperty(  
qa_const_string name  
qa_bool * value  
)`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_bool` value

Remarks Gets the value of the `qa_bool` message store property with the specified name.

Returns true if and only if the operation succeeded

## getByteStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::getByteStoreProperty(  
qa_const_string name  
qa_byte * value  
)`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_byte` value

Remarks Gets the value of the `qa_byte` message store property with the specified name.

Returns true if and only if the operation succeeded

## getDoubleStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::getDoubleStoreProperty(  
qa_const_string name  
qa_double * value  
)`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_double` value

Remarks Gets the value of the `qa_double` message store property with the specified name.

Returns true if and only if the operation succeeded

---

## getFloatStoreProperty Function

|            |                                                                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QAManagerBase::getFloatStoreProperty</b>(<br/>    qa_const_string <i>name</i><br/>    qa_float * <i>value</i><br/>)</pre>           |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>name</b> the name of the property to get</li><li>◆ <b>value</b> the destination for the qa_float value</li></ul> |
| Remarks    | Gets the value of the <code>qa_float</code> message store property with the specified name.                                                                 |
| Returns    | true if and only if the operation succeeded                                                                                                                 |

## getIntStoreProperty Function

|            |                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QAManagerBase::getIntStoreProperty</b>(<br/>    qa_const_string <i>name</i><br/>    qa_int * <i>value</i><br/>)</pre>             |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>name</b> the name of the property to get</li><li>◆ <b>value</b> the destination for the qa_int value</li></ul> |
| Remarks    | Gets the value of the <code>qa_int</code> message store property with the specified name.                                                                 |
| Returns    | true if and only if the operation succeeded                                                                                                               |

## getLastError Function

|          |                                                                |
|----------|----------------------------------------------------------------|
| Synopsis | <pre>virtual qa_int <b>QAManagerBase::getLastError</b>()</pre> |
| Remarks  | Gets the error code of the last method call that failed.       |
| Returns  | the error code                                                 |

## getLastErrorMsg Function

|          |                                                                      |
|----------|----------------------------------------------------------------------|
| Synopsis | <pre>virtual qa_string <b>QAManagerBase::getLastErrorMsg</b>()</pre> |
| Remarks  | Gets an error message corresponding to the error code.               |
| Returns  | the error message                                                    |

## getLongStoreProperty Function

|            |                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManagerBase::getLongStoreProperty</b> (<br>qa_const_string <i>name</i><br>qa_long * <i>value</i><br>)<br>                                    |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to get</li> <li>◆ <b>value</b>   the destination for the qa_long value</li> </ul> |
| Remarks    | Gets the value of the qa_long message store property with the specified name.                                                                                     |
| Returns    | true if and only if the operation succeeded                                                                                                                       |

## getMessage Function

|            |                                                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual QAMessage * <b>QAManagerBase::getMessage</b> (<br>qa_const_string <i>dest</i><br>)<br>                                  |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>dest</b>   the destination</li> </ul>                                               |
| Remarks    | Gets the next message that is queued for the given destination, waiting indefinitely if there are currently no messages queued. |
| Returns    | the next message, or qa_null if no message is available                                                                         |

## getMessageNoWait Function

|            |                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual QAMessage * <b>QAManagerBase::getMessageNoWait</b> (<br>qa_const_string <i>dest</i><br>)<br>                         |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>dest</b>   the destination</li> </ul>                                            |
| Remarks    | Gets the next message that is queued for the given destination, returning qa_null if there are currently no messages queued. |
| Returns    | the next message, or qa_null if no message is available                                                                      |

## getMessageTimeout Function

|            |                                                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual QAMessage * <b>QAManagerBase::getMessageTimeout</b> (<br>qa_const_string <i>dest</i><br>qa_long <i>timeout</i><br>)<br> |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>dest</b>   the destination</li> </ul>                                               |

---

|         |                                                                                                                                                 |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------|
|         | ◆ <b>timeout</b> the maximum time, in milliseconds, to wait                                                                                     |
| Remarks | Gets the next message that is queued for the given destination, waiting at most timeout milliseconds if there are currently no messages queued. |
| Returns | the next message, or qa_null if no message is available                                                                                         |

## getMode Function

|          |                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual qa_short <b>QAManagerBase::getMode()</b>                                                                                    |
| Remarks  | Gets the acknowledge mode of this instance of QAManagerBase. Mode is IMPLICIT_ACKNOWLEDGE or EXPLICIT_ACKNOWLEDGE or TRANSACTIONAL. |
| Returns  | the acknowledge mode                                                                                                                |
| See Also | <a href="#">Class AcknowledgementMode</a>                                                                                           |

## getShortStoreProperty Function

|            |                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManagerBase::getShortStoreProperty</b> (<br>qa_const_string <i>name</i><br>qa_short * <i>value</i><br>)<br> |
| Parameters | ◆ <b>name</b> the name of the property to get<br>◆ <b>value</b> the destination for the qa_short value                           |
| Remarks    | Gets the value of the qa_short message store property with the specified name.                                                   |
| Returns    | true if and only if the operation succeeded                                                                                      |

## getStringStoreProperty Function

|            |                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_int <b>QAManagerBase::getStringStoreProperty</b> (<br>qa_const_string <i>name</i><br>qa_string <i>dest</i><br>qa_int <i>maxlen</i><br>)<br>                                                               |
| Parameters | ◆ <b>name</b> the name of the property to get<br>◆ <b>dest</b> the destination for the qa_string value<br>◆ <b>maxlen</b> the maximum number of qa_chars of the value to copy, including the null terminator qa_char |



Remarks Gets the value of the message store property with the specified name.

Returns the number of non-null qa\_chars actually copied, or -1 if the operation failed

## peekFirstMessage Function

Synopsis `virtual QAMessage * QAManagerBase::peekFirstMessage( qa_const_string dest )`

Parameters ♦ **dest** the destination

Remarks Looks at the first message that is queued for the given destination, returning qa\_null if there are currently no messages queued. This method is used before peekNextMessage, which can be used to enumerate the messages queued for the given destination at the time this method was called.

Returns the next message, or qa\_null if no message is available

## peekNextMessage Function

Synopsis `virtual QAMessage * QAManagerBase::peekNextMessage()`

Remarks Looks at the next message that is queued for the given destination, returning qa\_null if there are currently no more messages queued. This method is used after peekFirstMessage, and can be used to enumerate the messages queued for the given destination at the time peekFirstMessage was called.

Returns the next message, or qa\_null if no message is available

## publishMessage Function

Synopsis `virtual qa_bool QAManagerBase::publishMessage( qa_const_string dest QAMessage * msg )`

Parameters ♦ **dest** the destination  
♦ **msg** the message

Remarks Not implemented.

## putMessage Function

Synopsis `virtual qa_bool QAManagerBase::putMessage( qa_const_string dest QAMessage * msg )`

---

|            |                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------|
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>dest</b> the destination</li> <li>◆ <b>msg</b> the message</li> </ul> |
| Remarks    | Puts a message into the queue for the given destination.                                                          |
| Returns    | true if and only if the operation succeeded                                                                       |

## putMessageTimeToLive Function

|            |                                                                                                                                                                           |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool QAManagerBase::putMessageTimeToLive(     qa_const_string dest     QAMessage * msg     qa_long ttl )</pre>                                            |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>dest</b> the destination</li> <li>◆ <b>msg</b> the message</li> <li>◆ <b>ttl</b> the time-to-live, in milliseconds</li> </ul> |
| Remarks    | Puts a message into the queue for the given destination and a given time-to-live in milliseconds.                                                                         |
| Returns    | true if and only if the operation succeeded                                                                                                                               |

## setBooleanStoreProperty Function

|            |                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool QAManagerBase::setBooleanStoreProperty(     qa_const_string name     qa_bool value )</pre>                                           |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to set</li> <li>◆ <b>value</b> the qa_bool value of the property</li> </ul> |
| Remarks    | Sets a qa_bool message store property value with the specified name.                                                                                      |
| Returns    | true if and only if the operation succeeded                                                                                                               |

## setByteStoreProperty Function

|            |                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool QAManagerBase::setByteStoreProperty(     qa_const_string name     qa_byte value )</pre>                                              |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to set</li> <li>◆ <b>value</b> the qa_byte value of the property</li> </ul> |

Remarks Sets a `qa_byte` message store property value with the specified name.  
Returns true if and only if the operation succeeded

## setDoubleStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::setDoubleStoreProperty(  
 qa_const_string name  
 qa_double value  
)`

Parameters ♦ **name** the name of the property to set  
♦ **value** the `qa_double` value of the property

Remarks Sets a `qa_double` message store property value with the specified name.  
Returns true if and only if the operation succeeded

## setFloatStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::setFloatStoreProperty(  
 qa_const_string name  
 qa_float value  
)`

Parameters ♦ **name** the name of the property to set  
♦ **value** the `qa_float` value of the property

Remarks Sets a `qa_float` message store property value with the specified name.  
Returns true if and only if the operation succeeded

## setIntStoreProperty Function

Synopsis `virtual qa_bool QAManagerBase::setIntStoreProperty(  
 qa_const_string name  
 qa_int value  
)`

Parameters ♦ **name** the name of the property to set  
♦ **value** the `qa_int` value of the property

Remarks Sets a `qa_int` message store property value with the specified name.  
Returns true if and only if the operation succeeded

---

## setLongStoreProperty Function

|            |                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QAManagerBase::setLongStoreProperty</b>(<br/>    qa_const_string <i>name</i><br/>    qa_long <i>value</i><br/>)</pre>          |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>name</b> the name of the property to set</li><li>◆ <b>value</b> the qa_long value of the property</li></ul> |
| Remarks    | Sets a qa_long message store property value with the specified name.                                                                                   |
| Returns    | true if and only if the operation succeeded                                                                                                            |

## setMessageListener Function

|            |                                                                                                                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual void <b>QAManagerBase::setMessageListener</b>(<br/>    qa_const_string <i>dest</i><br/>    QAMessageListener * <i>listener</i><br/>)</pre>                                                |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>dest</b> the destination address that the listener applies to.</li><li>◆ <b>listener</b> the message listener to associate with destination dest.</li></ul> |
| Remarks    | Sets the message listener associated with a destination.                                                                                                                                               |

## setProperty Function

|            |                                                                                                                                                                                                                                                  |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_bool <b>QAManagerBase::setProperty</b>(<br/>    qa_const_string <i>name</i><br/>    qa_const_string <i>value</i><br/>)</pre>                                                                                                     |
| Parameters | <ul style="list-style-type: none"><li>◆ <b>name</b> the name of the property to set</li><li>◆ <b>value</b> the value of the property</li></ul>                                                                                                   |
| Remarks    | Sets the named property to the given value. Properties for this QAManagerBase may be set with this method as an alternative to the properties file at creation. Properties must be set before calling the open() methods of the derived classes. |
| Returns    | true if and only if the operation succeeded                                                                                                                                                                                                      |

## setShortStoreProperty Function

|            |                                                                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManagerBase::setShortStoreProperty</b> (<br>qa_const_string <i>name</i><br>qa_short <i>value</i><br>)                                     |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_short value of the property</li> </ul> |
| Remarks    | Sets a qa_short message store property value with the specified name.                                                                                          |
| Returns    | true if and only if the operation succeeded                                                                                                                    |

## setStringStoreProperty Function

|            |                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAManagerBase::setStringStoreProperty</b> (<br>qa_const_string <i>name</i><br>qa_const_string <i>value</i><br>)                              |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_string value of the property</li> </ul> |
| Remarks    | Sets the named message store property to the given value.                                                                                                       |
| Returns    | true if and only if the operation succeeded                                                                                                                     |

## start Function

|          |                                                           |
|----------|-----------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAManagerBase::start</b> ()            |
| Remarks  | Starts the QAManagerBase for receiving incoming messages. |
| Returns  | true if and only if the operation succeeded               |

## stop Function

|          |                                                           |
|----------|-----------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAManagerBase::stop</b> ()             |
| Remarks  | Stops the QAManagerBase's reception of incoming messages. |
| Returns  | true if and only if the operation succeeded               |

## triggerSendReceive Function

|          |                                                             |
|----------|-------------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAManagerBase::triggerSendReceive</b> () |
|----------|-------------------------------------------------------------|

---

|         |                                                                                                                                                            |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Remarks | Causes any pending messages to be sent and received. This includes both messages queued locally, and messages queued on the server for local destinations. |
| Returns | true if and only if the operation succeeded                                                                                                                |

### **~QAManagerBase Function**

|          |                                                 |
|----------|-------------------------------------------------|
| Synopsis | virtual <b>QAManagerBase::~~QAManagerBase()</b> |
| Remarks  | Virtual destructor                              |

## Class QAManagerFactory

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | public <b>QAManagerFactory</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Remarks  | This class acts as a factory class for creating <a href="#">Class QATransactionalManager</a> and <a href="#">Class QAManager</a> objects.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Members  | All members of QAManagerFactory, including all inherited members. <ul style="list-style-type: none"> <li>◆ <a href="#">“createQAManager Function” on page 153</a></li> <li>◆ <a href="#">“createQATransactionalManager Function” on page 153</a></li> <li>◆ <a href="#">“deleteQAManager Function” on page 153</a></li> <li>◆ <a href="#">“deleteQATransactionalManager Function” on page 154</a></li> <li>◆ <a href="#">“getLastError Function” on page 154</a></li> <li>◆ <a href="#">“getLastErrorMsg Function” on page 154</a></li> <li>◆ <a href="#">“~QAManagerFactory Function” on page 154</a></li> </ul> |

### createQAManager Function

|            |                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual QAManager * <b>QAManagerFactory::createQAManager</b> (<br>qa_const_string <i>iniFile</i><br>) |
| Parameters | ◆ <b>iniFile</b> the path of the properties file                                                      |
| Remarks    | Returns a new instance of a <a href="#">Class QAManager</a> with specified properties.                |
| Returns    | the <a href="#">Class QAManager</a> instance                                                          |

### createQATransactionalManager Function

|            |                                                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual QATransactionalManager *<br><b>QAManagerFactory::createQATransactionalManager</b> (<br>qa_const_string <i>iniFile</i><br>) |
| Parameters | ◆ <b>iniFile</b> the path of the properties file                                                                                   |
| Remarks    | Returns a new instance of a <a href="#">Class QATransactionalManager</a> with specified properties.                                |
| Returns    | the <a href="#">Class QATransactionalManager</a> instance                                                                          |

### deleteQAManager Function

|            |                                                                                        |
|------------|----------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAManagerFactory::deleteQAManager</b> (<br>QAManager * <i>mgr</i><br>) |
| Parameters | ◆ <b>mgr</b> the <a href="#">“Class QAManager” on page 138</a> to be destroyed         |

---

Remarks Destroys a [Class QAManager](#), freeing its resources. It is not necessary to use this method, since all created QAManager's will be destroyed when QAnywhereFactory\_term() is called. It is provided as a convenience for when it is desirable to free resources in a timely manner.

## deleteQATransactionalManager Function

Synopsis virtual void **QAManagerFactory::deleteQATransactionalManager**(  
QATransactionalManager \* mgr  
)

Parameters ♦ **mgr** the “[Class QATransactionalManager](#)” on page 171 to be destroyed

Remarks Destroys a [Class QATransactionalManager](#), freeing its resources. It is not necessary to use this method, since all created QATransactionalManager's will be destroyed when QAnywhereFactory\_term() is called. It is provided as a convenience for when it is desirable to free resources in a timely manner.

## getLastError Function

Synopsis virtual qa\_int **QAManagerFactory::getLastError**()

Remarks Gets the error code of the last method call that failed.

Returns the error code

## getLastErrorMsg Function

Synopsis virtual qa\_string **QAManagerFactory::getLastErrorMsg**()

Remarks Gets an error message corresponding to the error code.

Returns the error message

## ~QAManagerFactory Function

Synopsis virtual **QAManagerFactory::~~QAManagerFactory**()

Remarks Virtual destructor



## Class QAMessage

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis        | <code>public QAMessage</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Derived classes | <ul style="list-style-type: none"> <li>◆ <a href="#">“Class QABinaryMessage” on page 127</a></li> <li>◆ <a href="#">“Class QATextMessage” on page 168</a></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Remarks         | The QAMessage interface is the root interface of all QAnywhere client messages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Members         | <p>All members of QAMessage, including all inherited members.</p> <ul style="list-style-type: none"> <li>◆ <a href="#">“castToBinaryMessage Function” on page 156</a></li> <li>◆ <a href="#">“castToTextMessage Function” on page 156</a></li> <li>◆ <a href="#">“clearProperties Function” on page 156</a></li> <li>◆ <a href="#">“DEFAULT_PRIORITY Variable” on page 156</a></li> <li>◆ <a href="#">“DEFAULT_TIME_TO_LIVE Variable” on page 156</a></li> <li>◆ <a href="#">“getAddress Function” on page 157</a></li> <li>◆ <a href="#">“getBooleanProperty Function” on page 157</a></li> <li>◆ <a href="#">“getByteProperty Function” on page 157</a></li> <li>◆ <a href="#">“getDoubleProperty Function” on page 157</a></li> <li>◆ <a href="#">“getExpiration Function” on page 158</a></li> <li>◆ <a href="#">“getFloatProperty Function” on page 158</a></li> <li>◆ <a href="#">“getInReplyToID Function” on page 158</a></li> <li>◆ <a href="#">“getIntProperty Function” on page 158</a></li> <li>◆ <a href="#">“getLongProperty Function” on page 159</a></li> <li>◆ <a href="#">“getMessageID Function” on page 159</a></li> <li>◆ <a href="#">“getPriority Function” on page 159</a></li> <li>◆ <a href="#">“getPropertyNames Function” on page 159</a></li> <li>◆ <a href="#">“getPropertyType Function” on page 160</a></li> <li>◆ <a href="#">“getRedelivered Function” on page 160</a></li> <li>◆ <a href="#">“getReplyToAddress Function” on page 160</a></li> <li>◆ <a href="#">“getShortProperty Function” on page 160</a></li> <li>◆ <a href="#">“getStringProperty Function” on page 161</a></li> <li>◆ <a href="#">“getStringProperty Function” on page 161</a></li> <li>◆ <a href="#">“getTimestamp Function” on page 161</a></li> <li>◆ <a href="#">“getTimestampAsString Function” on page 162</a></li> <li>◆ <a href="#">“propertyExists Function” on page 162</a></li> <li>◆ <a href="#">“setAddress Function” on page 162</a></li> <li>◆ <a href="#">“setBooleanProperty Function” on page 163</a></li> <li>◆ <a href="#">“setByteProperty Function” on page 163</a></li> <li>◆ <a href="#">“setDoubleProperty Function” on page 163</a></li> <li>◆ <a href="#">“setFloatProperty Function” on page 163</a></li> <li>◆ <a href="#">“setInReplyToID Function” on page 164</a></li> <li>◆ <a href="#">“setIntProperty Function” on page 164</a></li> </ul> |

- 
- ◆ “setLongProperty Function” on page 164
  - ◆ “setMessageID Function” on page 164
  - ◆ “setPriority Function” on page 165
  - ◆ “setRedelivered Function” on page 165
  - ◆ “setReplyToAddress Function” on page 165
  - ◆ “setShortProperty Function” on page 165
  - ◆ “setStringProperty Function” on page 166
  - ◆ “setTimestamp Function” on page 166
  - ◆ “~QAMessage Function” on page 166

## DEFAULT\_PRIORITY Variable

Synopsis                    `const qa_int QAMessage::DEFAULT_PRIORITY`

Remarks                    The default message priority.

## DEFAULT\_TIME\_TO\_LIVE Variable

Synopsis                    `const qa_long QAMessage::DEFAULT_TIME_TO_LIVE`

Remarks                    The default message time-to-live value.

## castToBinaryMessage Function

Synopsis                    `virtual QABinaryMessage * QAMessage::castToBinaryMessage()`

Remarks                    Casts this QAMessage to a [Class QABinaryMessage](#).

Returns                    a pointer to the [Class QABinaryMessage](#), or NULL if this message is not an instance of [Class QABinaryMessage](#).

## castToTextMessage Function

Synopsis                    `virtual QATextMessage * QAMessage::castToTextMessage()`

Remarks                    Casts this QAMessage to a [Class QATextMessage](#).

Returns                    a pointer to the [Class QATextMessage](#), or NULL if this message is not an instance of [Class QATextMessage](#).

## clearProperties Function

Synopsis                    `virtual void QAMessage::clearProperties()`

Remarks                    Clears a message’s properties. The message’s header fields and body are not cleared.

## getAddress Function

|          |                                                                                                                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual qa_const_string <b>QAMessage::getAddress()</b>                                                                                                                                                 |
| Remarks  | Gets the destination address for this message. When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method. |
| Returns  | the destination address                                                                                                                                                                                |

## getBooleanProperty Function

|            |                                                                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAMessage::getBooleanProperty</b> (<br>qa_const_string <i>name</i><br>qa_bool * <i>value</i><br>)                                          |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to get</li> <li>◆ <b>value</b> the destination for the qa_bool value</li> </ul> |
| Remarks    | Gets the value of the qa_bool property with the specified name.                                                                                               |
| Returns    | true if and only if the operation succeeded                                                                                                                   |

## getByteProperty Function

|            |                                                                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAMessage::getByteProperty</b> (<br>qa_const_string <i>name</i><br>qa_byte * <i>value</i><br>)                                             |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to get</li> <li>◆ <b>value</b> the destination for the qa_byte value</li> </ul> |
| Remarks    | Gets the value of the qa_byte property with the specified name.                                                                                               |
| Returns    | true if and only if the operation succeeded                                                                                                                   |

## getDoubleProperty Function

|            |                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAMessage::getDoubleProperty</b> (<br>qa_const_string <i>name</i><br>qa_double * <i>value</i><br>)                                           |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to get</li> <li>◆ <b>value</b> the destination for the qa_double value</li> </ul> |

---

Remarks Gets the value of the `qa_double` property with the specified name.

Returns true if and only if the operation succeeded

## getExpiration Function

Synopsis `virtual qa_long QAMessage::getExpiration()`

Remarks Gets the message's expiration value. When a message is sent, the Expiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish. If the time-to-live is specified as zero, Expiration is set to zero to indicate that the message does not expire.

Returns the expiration

## getFloatProperty Function

Synopsis `virtual qa_bool QAMessage::getFloatProperty(  
qa_const_string name  
qa_float * value  
)`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_float` value

Remarks Gets the value of the `qa_float` property with the specified name.

Returns true if and only if the operation succeeded

## getInReplyToID Function

Synopsis `virtual qa_const_string QAMessage::getInReplyToID()`

Remarks Gets the In-Reply-To ID for the message.

Returns the In-Reply-To ID

## getIntProperty Function

Synopsis `virtual qa_bool QAMessage::getIntProperty(  
qa_const_string name  
qa_int * value  
)`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_int` value

Remarks Gets the value of the `qa_int` property with the specified name.

Returns true if and only if the operation succeeded

## getLongProperty Function

Synopsis `virtual qa_bool QAMessage::getLongProperty( qa_const_string name qa_long * value )`

Parameters

- ◆ **name** the name of the property to get
- ◆ **value** the destination for the `qa_long` value

Remarks Gets the value of the `qa_long` property with the specified name.

Returns true if and only if the operation succeeded

## getMessageID Function

Synopsis `virtual qa_const_string QAMessage::getMessageID()`

Remarks Gets the message ID. The MessageID header field contains a value that uniquely identifies each message sent by the QAnywhere client. When a message is sent, MessageID can be ignored. When the send method returns, it contains an assigned value. A MessageID is a `qa_string` value that should function as a unique key for identifying messages in a historical repository.

Returns the message ID

## getPriority Function

Synopsis `virtual qa_int QAMessage::getPriority()`

Remarks Gets the message priority level. The QAnywhere client API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

Returns the priority

## getPropertyNames Function

Synopsis `virtual qa_const_string * QAMessage::getPropertyNames()`

Remarks Returns a list of property names currently set in the QAMessage.

Returns a NULL-terminated array of property names

---

## getPropertyType Function

|            |                                                                                                                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_short <b>QAMessage::getPropertyType</b> (<br>qa_const_string <i>name</i><br>)                                                                                                                                                                  |
| Parameters | ◆ <b>name</b> the name of the property                                                                                                                                                                                                                    |
| Remarks    | Returns the type of a property with the given name. One of PROPERTY_TYPE_BOOLEAN, PROPERTY_TYPE_BYTE, PROPERTY_TYPE_SHORT, PROPERTY_TYPE_INT, PROPERTY_TYPE_LONG, PROPERTY_TYPE_FLOAT, PROPERTY_TYPE_DOUBLE, PROPERTY_TYPE_STRING, PROPERTY_TYPE_UNKNOWN. |
| Returns    | the type of the property                                                                                                                                                                                                                                  |

## getRedelivered Function

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QAMessage::getRedelivered</b> ()              |
| Remarks  | Gets an indication of whether this message is being redelivered. |
| Returns  | whether the message was redelivered                              |

## getReplyToAddress Function

|          |                                                                   |
|----------|-------------------------------------------------------------------|
| Synopsis | virtual qa_const_string <b>QAMessage::getReplyToAddress</b> ()    |
| Remarks  | Gets the address to which a reply to this message should be sent. |
| Returns  | the Reply-To address                                              |

## getShortProperty Function

|            |                                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_bool <b>QAMessage::getShortProperty</b> (<br>qa_const_string <i>name</i><br>qa_short * <i>value</i><br>) |
| Parameters | ◆ <b>name</b> the name of the property to get<br>◆ <b>value</b> the destination for the qa_short value              |
| Remarks    | Gets the value of the qa_short property with the specified name.                                                    |
| Returns    | true if and only if the operation succeeded                                                                         |

## getStringProperty Function

|            |                                                                                                                                                                                                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_int <b>QAMessage::getStringProperty</b>(     qa_const_string <i>name</i>     qa_string <i>dest</i>     qa_int <i>maxlen</i> )</pre>                                                                                                                                |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to get</li> <li>◆ <b>dest</b> the destination for the qa_string value</li> <li>◆ <b>maxlen</b> the maximum number of qa_chars of the value to copy, including the null terminator qa_char</li> </ul> |
| Remarks    | Gets the value of the qa_string property with the specified name.                                                                                                                                                                                                                  |
| Returns    | the number of non-null qa_chars actually copied, or -1 if the operation failed                                                                                                                                                                                                     |

## getStringProperty Function

|            |                                                                                                                                                                                                                                                                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | <pre>virtual qa_int <b>QAMessage::getStringProperty</b>(     qa_const_string <i>name</i>     qa_int <i>offset</i>     qa_string <i>dest</i>     qa_int <i>maxlen</i> )</pre>                                                                                                                                                                                               |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b> the name of the property to get</li> <li>◆ <b>offset</b> the starting offset into the property value from which to copy</li> <li>◆ <b>dest</b> the destination for the qa_string value</li> <li>◆ <b>maxlen</b> the maximum number of qa_chars of the value to copy, including the null terminator qa_char</li> </ul> |
| Remarks    | Gets the value of the qa_string property (starting at offset) with the specified name.                                                                                                                                                                                                                                                                                     |
| Returns    | the number of non-null qa_chars actually copied, or -1 if the operation failed<br>Returns the value of the qa_string property (starting at offset) with the specified name.                                                                                                                                                                                                |

## getTimestamp Function

|          |                                                             |
|----------|-------------------------------------------------------------|
| Synopsis | <pre>virtual qa_long <b>QAMessage::getTimestamp</b>()</pre> |
|----------|-------------------------------------------------------------|

---

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Remarks | Gets the message timestamp. The Timestamp header field contains the time a message was created. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages. It is in units that are natural for the platform. For Windows/PocketPC platforms, the timestamp is the SYSTEMTIME, converted to a FILETIME, which is copied to an qa_long value. |
| Returns | the message timestamp                                                                                                                                                                                                                                                                                                                                                                                                                           |

## getTimestampAsString Function

Synopsis                    virtual qa\_int **QAMessage::getTimestampAsString**(  
                            qa\_string *buffer*  
                            qa\_int *bufferLen*  
                            )

Parameters

- ◆ **buffer**    the buffer for the formatted timestamp
- ◆ **bufferLen**    the size of the buffer

Remarks                    Gets the message timestamp as a formatted string. The format is: “dow, MMM dd, yyyy hh:mm:ss.nnn GMT”.

Returns                    the number of non-null qa\_char’s written to the buffer

## propertyExists Function

Synopsis                    virtual qa\_bool **QAMessage::propertyExists**(  
                            qa\_const\_string *name*  
                            )

Parameters

- ◆ **name**    the name of the property

Remarks                    Indicates whether a property value exists.

Returns                    true if and only if the property exists

## setAddress Function

Synopsis                    virtual void **QAMessage::setAddress**(  
                            qa\_const\_string *destination*  
                            )

Parameters

- ◆ **destination**    the destination address

Remarks                    Sets the destination address for this message. This method can be used to change the value for a message that has been received.



## setBooleanProperty Function

|            |                                                                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setBooleanProperty</b> (<br>qa_const_string <i>name</i><br>qa_bool <i>value</i><br>)<br>                                           |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_bool value of the property</li> </ul> |
| Remarks    | Sets a qa_bool property value with the specified name into the message.                                                                                       |

## setByteProperty Function

|            |                                                                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setByteProperty</b> (<br>qa_const_string <i>name</i><br>qa_byte <i>value</i><br>)<br>                                              |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_byte value of the property</li> </ul> |
| Remarks    | Sets a qa_byte property value with the specified name into the message.                                                                                       |

## setDoubleProperty Function

|            |                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setDoubleProperty</b> (<br>qa_const_string <i>name</i><br>qa_double <i>value</i><br>)<br>                                            |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_double value of the property</li> </ul> |
| Remarks    | Sets a qa_double property value with the specified name into the message.                                                                                       |

## setFloatProperty Function

|            |                                                                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setFloatProperty</b> (<br>qa_const_string <i>name</i><br>qa_float <i>value</i><br>)<br>                                             |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>name</b>   the name of the property to set</li> <li>◆ <b>value</b>   the qa_float value of the property</li> </ul> |

---

Remarks                      Sets a `qa_float` property value with the specified name into the message.

## setInReplyToID Function

Synopsis                      virtual void **QAMessage::setInReplyToID**(  
                                 `qa_const_string id`  
                                 )

Parameters                    ♦ **id**    the In-Reply-To ID

Remarks                      Sets the In-Reply-To ID for the message. A client can use the `InReplyToID` header field to link one message with another. A typical use is to link a response message with its request message.

## setIntProperty Function

Synopsis                      virtual void **QAMessage::setIntProperty**(  
                                 `qa_const_string name`  
                                 `qa_int value`  
                                 )

Parameters                    ♦ **name**    the name of the property to set  
                                 ♦ **value**    the `qa_int` value of the property

Remarks                      Sets a `qa_int` property value with the specified name into the message.

## setLongProperty Function

Synopsis                      virtual void **QAMessage::setLongProperty**(  
                                 `qa_const_string name`  
                                 `qa_long value`  
                                 )

Parameters                    ♦ **name**    the name of the property to set  
                                 ♦ **value**    the `qa_long` value of the property

Remarks                      Sets a `qa_long` property value with the specified name into the message.

## setMessageID Function

Synopsis                      virtual void **QAMessage::setMessageID**(  
                                 `qa_const_string id`  
                                 )

Parameters                    ♦ **id**    the message ID

Remarks                      Sets the message ID. This method can be used to change the value for a message that has been received.

## setPriority Function

|            |                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setPriority</b> (<br>qa_int <i>priority</i><br>)                                                 |
| Parameters | ◆ <b>priority</b> the priority                                                                                              |
| Remarks    | Sets the priority level for this message. This method can be used to change the value for a message that has been received. |

## setRedelivered Function

|            |                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setRedelivered</b> (<br>qa_bool <i>redelivered</i><br>)                                                            |
| Parameters | ◆ <b>redelivered</b> the redelivered indication                                                                                               |
| Remarks    | Sets an indication of whether this message was redelivered. This method can be used to change the value for a message that has been received. |

## setReplyToAddress Function

|            |                                                                                           |
|------------|-------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setReplyToAddress</b> (<br>qa_const_string <i>replyTo</i><br>) |
| Parameters | ◆ <b>replyTo</b> the Reply-To address                                                     |
| Remarks    | Sets the address to which a reply to this message should be sent.                         |

## setShortProperty Function

|            |                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setShortProperty</b> (<br>qa_const_string <i>name</i><br>qa_short <i>value</i><br>) |
| Parameters | ◆ <b>name</b> the name of the property to set<br>◆ <b>value</b> the qa_short value of the property             |
| Remarks    | Sets a qa_short property value with the specified name into the message.                                       |

---

## setStringProperty Function

|            |                                                                                                                        |
|------------|------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setStringProperty</b> (<br>qa_const_string <i>name</i><br>qa_const_string <i>value</i><br>) |
| Parameters | ◆ <b>name</b> the name of the property to set<br>◆ <b>value</b> the qa_string value of the property                    |
| Remarks    | Sets a qa_string property value with the specified name into the message.                                              |

## setTimestamp Function

|            |                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessage::setTimestamp</b> (<br>qa_long <i>timestamp</i><br>)                                |
| Parameters | ◆ <b>timestamp</b> the message timestamp                                                                      |
| Remarks    | Sets the message timestamp. This method can be used to change the value for a message that has been received. |

## ~QAMessage Function

|          |                                          |
|----------|------------------------------------------|
| Synopsis | virtual <b>QAMessage::~~QAMessage</b> () |
| Remarks  | Virtual destructor                       |

## Class QAMessageListener

|          |                                                                                                                                                                                                                                           |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | public <b>QAMessageListener</b>                                                                                                                                                                                                           |
| Remarks  | A QAMessageListener object is used to receive asynchronously delivered messages.                                                                                                                                                          |
| Members  | All members of QAMessageListener, including all inherited members. <ul style="list-style-type: none"><li>◆ <a href="#">“onMessage Function” on page 167</a></li><li>◆ <a href="#">“~QAMessageListener Function” on page 167</a></li></ul> |

### onMessage Function

|            |                                                                                       |
|------------|---------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QAMessageListener::onMessage</b> (<br>QAMessage * <i>message</i><br>) |
| Parameters | ◆ <b>message</b> the message passed to the listener                                   |
| Remarks    | Passes a message to the listener.                                                     |

### ~QAMessageListener Function

|          |                                                          |
|----------|----------------------------------------------------------|
| Synopsis | virtual <b>QAMessageListener::~~QAMessageListener</b> () |
| Remarks  | Virtual destructor                                       |

---

# Class QTextMessage

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | public <b>QTextMessage</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Base classes | ◆ “Class QAMessage” on page 155                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Remarks      | <p>A QTextMessage object is used to send a message containing an <code>qa_string</code>. It inherits from the <a href="#">Class QAMessage</a> class and adds a text message body.</p> <p>When a client receives an QTextMessage, it is in read-only mode. If a client attempts to write to the message at this point, a <code>COMMON_MSG_NOT_WRITEABLE_ERROR</code> is set.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Members      | <p>All members of QTextMessage, including all inherited members.</p> <ul style="list-style-type: none"><li>◆ “castToBinaryMessage Function” on page 156</li><li>◆ “castToTextMessage Function” on page 156</li><li>◆ “clearProperties Function” on page 156</li><li>◆ “DEFAULT_PRIORITY Variable” on page 156</li><li>◆ “DEFAULT_TIME_TO_LIVE Variable” on page 156</li><li>◆ “getAddress Function” on page 157</li><li>◆ “getBooleanProperty Function” on page 157</li><li>◆ “getByteProperty Function” on page 157</li><li>◆ “getDoubleProperty Function” on page 157</li><li>◆ “getExpiration Function” on page 158</li><li>◆ “getFloatProperty Function” on page 158</li><li>◆ “getInReplyToID Function” on page 158</li><li>◆ “getIntProperty Function” on page 158</li><li>◆ “getLongProperty Function” on page 159</li><li>◆ “getMessageID Function” on page 159</li><li>◆ “getPriority Function” on page 159</li><li>◆ “getPropertyNames Function” on page 159</li><li>◆ “getPropertyType Function” on page 160</li><li>◆ “getRedelivered Function” on page 160</li><li>◆ “getReplyToAddress Function” on page 160</li><li>◆ “getShortProperty Function” on page 160</li><li>◆ “getStringProperty Function” on page 161</li><li>◆ “getStringProperty Function” on page 161</li><li>◆ “getText Function” on page 169</li><li>◆ “getTextLength Function” on page 169</li><li>◆ “getTimestamp Function” on page 161</li><li>◆ “getTimestampAsString Function” on page 162</li><li>◆ “propertyExists Function” on page 162</li><li>◆ “readText Function” on page 169</li><li>◆ “setAddress Function” on page 162</li></ul> |

- ◆ “setBooleanProperty Function” on page 163
- ◆ “setByteProperty Function” on page 163
- ◆ “setDoubleProperty Function” on page 163
- ◆ “setFloatProperty Function” on page 163
- ◆ “setInReplyToID Function” on page 164
- ◆ “setIntProperty Function” on page 164
- ◆ “setLongProperty Function” on page 164
- ◆ “setMessageID Function” on page 164
- ◆ “setPriority Function” on page 165
- ◆ “setRedelivered Function” on page 165
- ◆ “setReplyToAddress Function” on page 165
- ◆ “setShortProperty Function” on page 165
- ◆ “setStringProperty Function” on page 166
- ◆ “setText Function” on page 170
- ◆ “setTimestamp Function” on page 166
- ◆ “writeText Function” on page 170
- ◆ “~QAMessage Function” on page 166
- ◆ “~QATextMessage Function” on page 170

## getText Function

|          |                                                                            |
|----------|----------------------------------------------------------------------------|
| Synopsis | virtual qa_string <b>QATextMessage::getText()</b>                          |
| Remarks  | Gets the string containing this message’s data. The default value is null. |
| Returns  | the qa_string containing the message’s data                                |

## getTextLength Function

|          |                                                                                                                                                                                                                               |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual qa_long <b>QATextMessage::getTextLength()</b>                                                                                                                                                                         |
| Remarks  | Returns the text length. NOTE: If the text length is non-zero and <a href="#">getText Function</a> returns qa_null then the text does not fit in memory, and must be read in pieces using <a href="#">readText Function</a> . |

## readText Function

|            |                                                                                                                                                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual qa_int <b>QATextMessage::readText</b> (<br>qa_string <i>string</i><br>qa_int <i>length</i><br>)                                                                                                                            |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>string</b> the destination for the text</li> <li>◆ <b>length</b> the maximum number of qa_chars to read into the destination buffer, including the null termination qa_char</li> </ul> |

---

|         |                                                                                                              |
|---------|--------------------------------------------------------------------------------------------------------------|
| Remarks | Reads the requested length of text from the current text position into a buffer.                             |
| Returns | the actual number of non-null qa_chars read, or -1 if the current text position is after the end of the text |

## setText Function

|            |                                                                                    |
|------------|------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QATextMessage::setText</b> (<br>qa_const_string <i>string</i><br>) |
| Parameters | ◆ <b>string</b> the qa_string containing the message's data                        |
| Remarks    | Sets the string containing this message's data.                                    |

## writeText Function

|            |                                                                                                                                                                                                  |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis   | virtual void <b>QATextMessage::writeText</b> (<br>qa_const_string <i>string</i><br>qa_int <i>offset</i><br>qa_int <i>length</i><br>)                                                             |
| Parameters | ◆ <b>string</b> the source text to concatenate<br>◆ <b>offset</b> the offset into the source text at which to start reading<br>◆ <b>length</b> the number of qa_chars of the source text to read |
| Remarks    | Concatenates text to the current text.                                                                                                                                                           |

## ~QATextMessage Function

|          |                                                 |
|----------|-------------------------------------------------|
| Synopsis | virtual <b>QATextMessage::~QATextMessage</b> () |
| Remarks  | Virtual destructor                              |



# Class QATransactionalManager

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis     | <code>public QATransactionalManager</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Base classes | ◆ “Class QAManagerBase” on page 141                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Remarks      | This class is the manager for transactional messaging.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Members      | All members of QATransactionalManager, including all inherited members. <ul style="list-style-type: none"> <li>◆ “close Function” on page 142</li> <li>◆ “commit Function” on page 172</li> <li>◆ “createBinaryMessage Function” on page 142</li> <li>◆ “createTextMessage Function” on page 142</li> <li>◆ “deleteMessage Function” on page 142</li> <li>◆ “getBooleanStoreProperty Function” on page 143</li> <li>◆ “getByteStoreProperty Function” on page 143</li> <li>◆ “getDoubleStoreProperty Function” on page 143</li> <li>◆ “getFloatStoreProperty Function” on page 144</li> <li>◆ “getIntStoreProperty Function” on page 144</li> <li>◆ “getLastError Function” on page 144</li> <li>◆ “getLastErrorMsg Function” on page 144</li> <li>◆ “getLongStoreProperty Function” on page 145</li> <li>◆ “getMessage Function” on page 145</li> <li>◆ “getMessageNoWait Function” on page 145</li> <li>◆ “getMessageTimeout Function” on page 145</li> <li>◆ “getMode Function” on page 146</li> <li>◆ “getShortStoreProperty Function” on page 146</li> <li>◆ “getStringStoreProperty Function” on page 146</li> <li>◆ “open Function” on page 172</li> <li>◆ “peekFirstMessage Function” on page 147</li> <li>◆ “peekNextMessage Function” on page 147</li> <li>◆ “publishMessage Function” on page 147</li> <li>◆ “putMessage Function” on page 147</li> <li>◆ “putMessageTimeToLive Function” on page 148</li> <li>◆ “rollback Function” on page 172</li> <li>◆ “setBooleanStoreProperty Function” on page 148</li> <li>◆ “setByteStoreProperty Function” on page 148</li> <li>◆ “setDoubleStoreProperty Function” on page 149</li> <li>◆ “setFloatStoreProperty Function” on page 149</li> <li>◆ “setIntStoreProperty Function” on page 149</li> <li>◆ “setLongStoreProperty Function” on page 150</li> <li>◆ “setMessageListener Function” on page 150</li> <li>◆ “setProperty Function” on page 150</li> <li>◆ “setShortStoreProperty Function” on page 151</li> </ul> |

- 
- ◆ “setStringStoreProperty Function” on page 151
  - ◆ “start Function” on page 151
  - ◆ “stop Function” on page 151
  - ◆ “triggerSendReceive Function” on page 151
  - ◆ “QAManagerBase Function” on page 152
  - ◆ “QATransactionalManager Function” on page 172

## commit Function

|          |                                                                                                                                             |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QATransactionalManager::commit()</b>                                                                                     |
| Remarks  | Commits the current transaction and begins a new transaction. The first transaction begins with the call to <a href="#">open Function</a> . |
| Returns  | true if and only if the operation was successful                                                                                            |

## open Function

|          |                                                       |
|----------|-------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QATransactionalManager::open()</b> |
| Remarks  | Opens the QATransactionalManager.                     |
| Returns  | true if and only if the operation was successful      |

## rollback Function

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Synopsis | virtual qa_bool <b>QATransactionalManager::rollback()</b>        |
| Remarks  | Rolls back the current transaction and begins a new transaction. |
| Returns  | true if and only if the operation was successful                 |

## ~QATransactionalManager Function

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Synopsis | virtual <b>QATransactionalManager::~QATransactionalManager()</b> |
| Remarks  | Virtual destructor                                               |

---

## CHAPTER 9

# iAnywhere.QAnywhere.Client namespace

About this chapter

The iAnywhere.QAnywhere.Client namespace contains classes and enumerations for building applications that handle QAnywhere messages.

Contents

| <b>Topic:</b>                                          | <b>page</b> |
|--------------------------------------------------------|-------------|
| <a href="#">AcknowledgementMode enumeration</a>        | 174         |
| <a href="#">MessageProperties class</a>                | 175         |
| <a href="#">MessageType enumeration</a>                | 180         |
| <a href="#">QABinaryMessage class</a>                  | 181         |
| <a href="#">QAEException class</a>                     | 192         |
| <a href="#">QAManager class</a>                        | 195         |
| <a href="#">QAManagerBase class</a>                    | 201         |
| <a href="#">QAManagerBase.MessageListener delegate</a> | 219         |
| <a href="#">QAManagerFactory class</a>                 | 220         |
| <a href="#">QAMessage class</a>                        | 224         |
| <a href="#">QAPROPERTYTYPE enumeration</a>             | 237         |
| <a href="#">QATextMessage class</a>                    | 238         |
| <a href="#">QATransactionalManager class</a>           | 242         |

---

# AcknowledgementMode enumeration

The acknowledgement modes for QAManager instances

Prototypes

```
' Visual Basic  
Public Enum AcknowledgementMode
```

```
// C#  
public enum AcknowledgementMode
```

Members

| Member                        | Description                                                                                                                                                                                                                                                                              |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EXPLICIT_-<br>ACKNOWLEDGEMENT | Indicates that messages are not acknowledged as received until a call to one of the manager acknowledge methods is made.                                                                                                                                                                 |
| IMPLICIT_-<br>ACKNOWLEDGEMENT | Indicates that all messages are acknowledged as received as soon as the getMessage returns to the caller. Similarly, for message listeners, the message is acknowledged as soon as the call to the message listener delegate returns.                                                    |
| TRANSACTIONAL                 | All message puts and gets done via this mode are done transactionally. That is, all puts and gets occur within a transaction and are all committed or rolled back together. There is always a transaction. Committing or rolling back a transaction implicitly begins a new transaction. |

## MessageProperties class

Standard message property names

Prototypes

```

Visual Basic
Public Class MessageProperties

C#
public class MessageProperties

```

### MessageProperties members

Public static fields  
(Shared)

| Member                                      | Description                                                                                                                                                                      |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ABS_RETRY_TIMEOUT</a> field     | Optional property for messages sent through a connector. The time at which send retries through the connector will be stopped and the send is failed                             |
| <a href="#">ADAPTER</a> field               | For “system” queue messages, a delimited list of network adapters that can be used to connect to the QAnywhere server.                                                           |
| <a href="#">COMPRESSED</a> field            | Indicates whether the message content is compressed.                                                                                                                             |
| <a href="#">FROM_ADDR</a> field             | Optional property indicating the address of the sender                                                                                                                           |
| <a href="#">MSG_TYPE</a> field              | Optional property indicating the type of the message.                                                                                                                            |
| <a href="#">NETWORK</a> field               | For “system” queue messages, a delimited list of network names that can be used to connect to the QAnywhere server.                                                              |
| <a href="#">NETWORK_STATUS</a> field        | For “system” queue messages, the state of the network connection. Value is 1 if connected, 0 otherwise.                                                                          |
| <a href="#">RETRY_FAILED</a> field          | Set by the connector when sending a message to the RetryFailedAddress. The receiving client can use this property to identify messages for which re-sending failed.              |
| <a href="#">RETRY_FAILED_ADDR</a> field     | Optional property for messages sent through a connector. Once either the RetryMax or RetryTimeout is exceeded, if this property is set, the message will be sent to this address |
| <a href="#">RETRY_FAILED_PRIORITY</a> field | Optional property for messages sent through a connector. If a message is sent to the RetryFailedAddress, the message priority will be set to this                                |
| <a href="#">RETRY_MAX</a> field             | Optional property for messages sent through a connector. The maximum number of send retries at the connector before failing the send                                             |

---

| Member                              | Description                                                                                                                                                 |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">RETRY_TIMEOUT field</a> | Optional property for messages sent through a connector. The duration after which send retries through the connector will be stopped and the send is failed |

Public instance  
constructors

| Member                                        | Description                                                |
|-----------------------------------------------|------------------------------------------------------------|
| <a href="#">MessageProperties constructor</a> | Initializes a new instance of the MessageProperties class. |

## MessageProperties constructor

Initializes a new instance of the MessageProperties class.

Prototypes

```

' Visual Basic
Public Sub New()

// C#
public MessageProperties();

```

## ABS\_RETRY\_TIMEOUT field

Optional property for messages sent through a connector. The time at which send retries through the connector will be stopped and the send is failed

Prototypes

```

' Visual Basic
Public Shared ABS_RETRY_TIMEOUT As String

// C#
public const string ABS_RETRY_TIMEOUT;

```

## ADAPTER field

For “system” queue messages, a delimited list of network adapters that can be used to connect to the QAnywhere server.

Prototypes

```

' Visual Basic
Public Shared ADAPTER As String

// C#
public const string ADAPTER;

```

## COMPRESSED field

Indicates whether the message content is compressed.

Prototypes

```
' Visual Basic
Public Shared COMPRESSED As String

// C#
public const string COMPRESSED;
```

## FROM\_ADDR field

Optional property indicating the address of the sender

Prototypes

```
' Visual Basic
Public Shared FROM_ADDR As String

// C#
public const string FROM_ADDR;
```

## MSG\_TYPE field

Optional property indicating the type of the message.

Prototypes

```
' Visual Basic
Public Shared MSG_TYPE As String

// C#
public const string MSG_TYPE;
```

## NETWORK field

For “system” queue messages, a delimited list of network names that can be used to connect to the QAnywhere server.

Prototypes

```
' Visual Basic
Public Shared NETWORK As String

// C#
public const string NETWORK;
```

## NETWORK\_STATUS field

For “system” queue messages, the state of the network connection. Value is 1 if connected, 0 otherwise.

Prototypes

```
' Visual Basic
Public Shared NETWORK_STATUS As String
```

---

```
// C#  
public const string NETWORK_STATUS;
```

## RETRY\_FAILED field

Set by the connector when sending a message to the RetryFailedAddress. The receiving client can use this property to identify messages for which re-sending failed.

Prototypes

```
· Visual Basic  
Public Shared RETRY_FAILED As String
```

```
// C#  
public const string RETRY_FAILED;
```

## RETRY\_FAILED\_ADDR field

Optional property for messages sent through a connector. Once either the RetryMax or RetryTimeout is exceeded, if this property is set, the message will be sent to this address

Prototypes

```
· Visual Basic  
Public Shared RETRY_FAILED_ADDR As String
```

```
// C#  
public const string RETRY_FAILED_ADDR;
```

## RETRY\_FAILED\_PRIORITY field

Optional property for messages sent through a connector. If a message is sent to the RetryFailedAddress, the message priority will be set to this

Prototypes

```
· Visual Basic  
Public Shared RETRY_FAILED_PRIORITY As String
```

```
// C#  
public const string RETRY_FAILED_PRIORITY;
```

## RETRY\_MAX field

Optional property for messages sent through a connector. The maximum number of send retries at the connector before failing the send

Prototypes

```
· Visual Basic  
Public Shared RETRY_MAX As String
```

```
// C#  
public const string RETRY_MAX;
```



## RETRY\_TIMEOUT field

Optional property for messages sent through a connector. The duration after which send retries through the connector will be stopped and the send is failed

Prototypes

**Visual Basic**

Public Shared **RETRY\_TIMEOUT** As String

**C#**

public const string **RETRY\_TIMEOUT**;

---

# MessageType enumeration

Valid values for the message type property of a message

Prototypes

```
' Visual Basic  
Public Enum MessageType
```

```
// C#  
public enum MessageType
```

Members

| Member                      | Description                                                                                                                                             |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| NETWORK_STATUS_NOTIFICATION | A message type indicating a message that notifies the receiver of a change to the network status of the current device.                                 |
| PUSH_NOTIFICATION           | A message type indicating a message that notifies the receiver of one or more messages ready for synchronizing from the message server.                 |
| REGULAR                     | If no message type property exists then the message type is assumed to be REGULAR. This type of message is not treated specially by the message system. |

## QABinaryMessage class

Encapsulation of a binary message

Prototypes

```

' Visual Basic
Public Class QABinaryMessage
    Inherits QAMessage

// C#
public class QABinaryMessage :
    QAMessage
  
```

### QABinaryMessage members

Public instance  
properties

| Member                                                             | Description                                                                                                                  |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Address property</a> (inherited from QAMessage)        | The address of this message. May be null, but is null never null in a message received via getMessage or a message listener. |
| <a href="#">BodyLength property</a>                                | The length in bytes of the message content                                                                                   |
| <a href="#">Expiration property</a> (inherited from QAMessage)     | Indicates the time after which the message may expire and be removed from the message system if it has not been received.    |
| <a href="#">InReplyToID property</a> (inherited from QAMessage)    | The message id of the message for which this message is a reply. May be null.                                                |
| <a href="#">MessageID property</a> (inherited from QAMessage)      | The globally unique message id of the message. This property is null until a message is put.                                 |
| <a href="#">Priority property</a> (inherited from QAMessage)       | The priority of the message (ranging from 0 to 9)                                                                            |
| <a href="#">Redelivered property</a> (inherited from QAMessage)    | Indicates whether the message has been previously received but not acknowledged.                                             |
| <a href="#">ReplyToAddress property</a> (inherited from QAMessage) | The replyTo address of this message. May be null.                                                                            |
| <a href="#">Timestamp property</a> (inherited from QAMessage)      | The message timestamp.                                                                                                       |

Public instance methods

| Member                                                               | Description                                                                                                 |
|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <a href="#">ClearProperties method</a> (inherited from QAMessage)    | Clear all the properties of the message                                                                     |
| <a href="#">GetBooleanProperty method</a> (inherited from QAMessage) | Gets a boolean message property                                                                             |
| <a href="#">GetDoubleProperty method</a> (inherited from QAMessage)  | Gets a double message property                                                                              |
| <a href="#">GetFloatProperty method</a> (inherited from QAMessage)   | Gets a float message property                                                                               |
| <a href="#">GetIntProperty method</a> (inherited from QAMessage)     | Gets an int message property                                                                                |
| <a href="#">GetLongProperty method</a> (inherited from QAMessage)    | Gets a long message property                                                                                |
| <a href="#">GetProperty method</a> (inherited from QAMessage)        | Gets a message property                                                                                     |
| <a href="#">GetPropertyNames method</a> (inherited from QAMessage)   | Gets an enumerator over the property names of the message                                                   |
| <a href="#">GetPropertyType method</a> (inherited from QAMessage)    | Returns the property type of the given property                                                             |
| <a href="#">GetSbyteProperty method</a> (inherited from QAMessage)   | Gets a signed byte message property                                                                         |
| <a href="#">GetShortProperty method</a> (inherited from QAMessage)   | Gets a short message property                                                                               |
| <a href="#">GetStringProperty method</a> (inherited from QAMessage)  | Gets a string message property                                                                              |
| <a href="#">PropertyExists method</a> (inherited from QAMessage)     | Indicates whether the given property has been set for this message                                          |
| <a href="#">ReadBinary method</a>                                    | Read from the beginning of the unread part of the binary value the given number of bytes into a byte array. |
| <a href="#">ReadBoolean method</a>                                   | Read from the beginning of the unread part of the binary value as a boolean.                                |
| <a href="#">ReadChar method</a>                                      | Read from the beginning of the unread part of the binary value as a char.                                   |
| <a href="#">ReadDouble method</a>                                    | Read from the beginning of the unread part of the binary value as a double.                                 |

| Member                                                                             | Description                                                                                                |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <a href="#">ReadFloat method</a>                                                   | Read from the beginning of the unread part of the binary value as a float.                                 |
| <a href="#">ReadInt method</a>                                                     | Read from the beginning of the unread part of the binary value as an int.                                  |
| <a href="#">ReadLong method</a>                                                    | Read from the beginning of the unread part of the binary value as a long.                                  |
| <a href="#">ReadSbyte method</a>                                                   | Read from the beginning of the unread part of the binary value as a signed byte.                           |
| <a href="#">ReadShort method</a>                                                   | Read from the beginning of the unread part of the binary value as a short.                                 |
| <a href="#">ReadString method</a>                                                  | Read from the beginning of the unread part of the binary value as a string.                                |
| <a href="#">Reset method</a>                                                       | Reset the message so that reading of values starts from the beginning of the message bytes                 |
| <a href="#">SetBooleanProperty method</a> (inherited from <code>QAMessage</code> ) | Sets a boolean property                                                                                    |
| <a href="#">SetDoubleProperty method</a> (inherited from <code>QAMessage</code> )  | Sets a double property                                                                                     |
| <a href="#">SetFloatProperty method</a> (inherited from <code>QAMessage</code> )   | Sets a float property                                                                                      |
| <a href="#">SetIntProperty method</a> (inherited from <code>QAMessage</code> )     | Sets an int property                                                                                       |
| <a href="#">SetLongProperty method</a> (inherited from <code>QAMessage</code> )    | Sets a long property                                                                                       |
| <a href="#">SetProperty method</a> (inherited from <code>QAMessage</code> )        | Sets a property. The property type must be one of the acceptable primitive types, or <code>String</code> . |
| <a href="#">SetSbyteProperty method</a> (inherited from <code>QAMessage</code> )   | Sets a byte property                                                                                       |
| <a href="#">SetShortProperty method</a> (inherited from <code>QAMessage</code> )   | Sets a short property                                                                                      |
| <a href="#">SetStringProperty method</a> (inherited from <code>QAMessage</code> )  | Sets a string property                                                                                     |
| <a href="#">WriteBinary method</a>                                                 | Append the byte array value to the bytes of this message.                                                  |

| Member                              | Description                                                                 |
|-------------------------------------|-----------------------------------------------------------------------------|
| <a href="#">WriteBoolean method</a> | Binary code, and append the boolean value to the bytes of this message.     |
| <a href="#">WriteChar method</a>    | Binary code, and append the char value to the bytes of this message.        |
| <a href="#">WriteDouble method</a>  | Binary code, and append the double value to the bytes of this message.      |
| <a href="#">WriteFloat method</a>   | Binary code, and append the float value to the bytes of this message.       |
| <a href="#">WriteInt method</a>     | Binary code, and append the int value to the bytes of this message.         |
| <a href="#">WriteLong method</a>    | Binary code, and append the long value to the bytes of this message.        |
| <a href="#">WriteSbyte method</a>   | Binary code, and append the signed byte value to the bytes of this message. |
| <a href="#">WriteShort method</a>   | Binary code, and append the short value to the bytes of this message.       |
| <a href="#">WriteString method</a>  | Binary code, and append the string value to the bytes of this message.      |

Protected instance methods

| Member                                                                  | Description                        |
|-------------------------------------------------------------------------|------------------------------------|
| <a href="#">Dispose method</a> (inherited from <code>QAMessage</code> ) | Clean up any resources being used. |

## BodyLength property

The length in bytes of the message content

Prototypes

**Visual Basic**  
Public Readonly Property **BodyLength** As Long

**C#**  
public long **BodyLength** {get;}

## ReadBinary method

Read from the beginning of the unread part of the binary value the given number of bytes into a byte array.

|              |                                                                                                                                                                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>Visual Basic</b> Public Function <b>ReadBinary</b>( _     ByVal <i>bytes</i> As Byte(), _     ByVal <i>len</i> As Integer _ ) As Integer  <b>C#</b> public int <b>ReadBinary</b>(     byte[] <i>bytes</i>,     int <i>len</i> ); </pre> |
| Parameters   | <ul style="list-style-type: none"> <li>◆ <b>bytes</b> the byte array that will contain the read bytes</li> <li>◆ <b>len</b> the maximum number of bytes to read</li> </ul>                                                                       |
| Return value | the number of bytes read                                                                                                                                                                                                                         |
| Exceptions   | <ul style="list-style-type: none"> <li>◆ <a href="#">QAException class</a> - if there was a conversion error reading the value or if there is no more input</li> </ul>                                                                           |

## ReadBoolean method

Read from the beginning of the unread part of the binary value as a boolean.

|              |                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>Visual Basic</b> Public Function <b>ReadBoolean</b>() As Boolean  <b>C#</b> public bool <b>ReadBoolean</b>(); </pre>                                          |
| Return value | the boolean value read                                                                                                                                                 |
| Exceptions   | <ul style="list-style-type: none"> <li>◆ <a href="#">QAException class</a> - if there was a conversion error reading the value or if there is no more input</li> </ul> |

## ReadChar method

Read from the beginning of the unread part of the binary value as a char.

|              |                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>Visual Basic</b> Public Function <b>ReadChar</b>() As Char  <b>C#</b> public char <b>ReadChar</b>(); </pre>                                                   |
| Return value | the character value read                                                                                                                                               |
| Exceptions   | <ul style="list-style-type: none"> <li>◆ <a href="#">QAException class</a> - if there was a conversion error reading the value or if there is no more input</li> </ul> |

---

## ReadDouble method

Read from the beginning of the unread part of the binary value as a double.

Prototypes

```
' Visual Basic  
Public Function ReadDouble() As Double
```

```
// C#  
public double ReadDouble();
```

Return value

the double value read

Exceptions

◆ [QAException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadFloat method

Read from the beginning of the unread part of the binary value as a float.

Prototypes

```
' Visual Basic  
Public Function ReadFloat() As Single
```

```
// C#  
public float ReadFloat();
```

Return value

the float value read

Exceptions

◆ [QAException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadInt method

Read from the beginning of the unread part of the binary value as an int.

Prototypes

```
' Visual Basic  
Public Function ReadInt() As Integer
```

```
// C#  
public int ReadInt();
```

Return value

the int value read

Exceptions

◆ [QAException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadLong method

Read from the beginning of the unread part of the binary value as a long.

Prototypes

```
' Visual Basic  
Public Function ReadLong() As Long
```



```
// C#  
public long ReadLong();
```

Return value

the long value read

Exceptions

- ◆ [QAEException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadSbyte method

Read from the beginning of the unread part of the binary value as a signed byte.

Prototypes

```
' Visual Basic  
Public Function ReadSbyte() As System.SByte
```

```
// C#  
public System.Sbyte ReadSbyte();
```

Return value

the signed byte value read

Exceptions

- ◆ [QAEException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadShort method

Read from the beginning of the unread part of the binary value as a short.

Prototypes

```
' Visual Basic  
Public Function ReadShort() As Short
```

```
// C#  
public short ReadShort();
```

Return value

the short value read

Exceptions

- ◆ [QAEException class](#) - if there was a conversion error reading the value or if there is no more input

## ReadString method

Read from the beginning of the unread part of the binary value as a string.

Prototypes

```
' Visual Basic  
Public Function ReadString() As String
```

```
// C#  
public string ReadString();
```

Return value

the string value read

Exceptions

- 
- ◆ [QAEException class](#) - if there was a conversion error reading the value or if there is no more input

## Reset method

Reset the message so that reading of values starts from the beginning of the message bytes

Prototypes

```
' Visual Basic
Public Sub Reset()

// C#
public void Reset();
```

## WriteBinary method

Append the byte array value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteBinary( _
    ByVal val As Byte(), _
    ByVal offset As Integer, _
    ByVal len As Integer _
)

// C#
public void WriteBinary(
    byte[] val,
    int offset,
    int len
);
```

Parameters

- ◆ **val** the byte array value
- ◆ **len** the number of bytes to write
- ◆ **offset** the byte array offset to begin writing

## WriteBoolean method

Binary code, and append the boolean value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteBoolean( _
    ByVal val As Boolean _
)

// C#
public void WriteBoolean(
    bool val
);
```

Parameters           ◆ **val**   the boolean value

## WriteChar method

Binary code, and append the char value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteChar( _
    ByVal val As Char _
)

// C#
public void WriteChar(
    char val
);
```

Parameters           ◆ **val**   the char value

## WriteDouble method

Binary code, and append the double value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteDouble( _
    ByVal val As Double _
)

// C#
public void WriteDouble(
    double val
);
```

Parameters           ◆ **val**   the double value

## WriteFloat method

Binary code, and append the float value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteFloat( _
    ByVal val As Single _
)

// C#
public void WriteFloat(
    float val
);
```

Parameters           ◆ **val**   the float value

---

## WriteInt method

Binary code, and append the int value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteInt( _
    ByVal val As Integer _
)

// C#
public void WriteInt(
    int val
);
```

Parameters

◆ **val** the int value

## WriteLong method

Binary code, and append the long value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteLong( _
    ByVal val As Long _
)

// C#
public void WriteLong(
    long val
);
```

Parameters

◆ **val** the long value

## WriteSbyte method

Binary code, and append the signed byte value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteSbyte( _
    ByVal val As System.SByte _
)

// C#
public void WriteSbyte(
    System.Sbyte val
);
```

Parameters

◆ **val** the signed byte value

## WriteShort method

Binary code, and append the short value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteShort( _
    ByVal val As Short _
)

// C#
public void WriteShort(
    short val
);
```

Parameters

◆ **val** the short value

## WriteString method

Binary code, and append the string value to the bytes of this message.

Prototypes

```
' Visual Basic
Public Sub WriteString( _
    ByVal val As String _
)

// C#
public void WriteString(
    string val
);
```

Parameters

◆ **val** the string value

---

# QAException class

Exception thrown by QAnywhere

Prototypes

```
' Visual Basic
Public Class QAException
    Inherits ApplicationException

// C#
public class QAException :
    ApplicationException
```

## QAException members

Public instance  
constructors

| Member                                  | Description          |
|-----------------------------------------|----------------------|
| <a href="#">QAException constructor</a> | Create a QAException |
| <a href="#">QAException constructor</a> | Create a QAException |

Public instance  
properties

| Member                                                    | Description                                                                                                |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <a href="#">ErrorCode property</a>                        | The error code of the exception                                                                            |
| <a href="#">HelpLink</a> (inherited from Exception)       | Gets or sets a link to the help file associated with this exception.                                       |
| <a href="#">InnerException</a> (inherited from Exception) | Gets the <a href="#">System.Exception</a> instance that caused the current exception.                      |
| <a href="#">Message</a> (inherited from Exception)        | Gets a message that describes the current exception.                                                       |
| <a href="#">Source</a> (inherited from Exception)         | Gets or sets the name of the application or the object that causes the error.                              |
| <a href="#">StackTrace</a> (inherited from Exception)     | Gets a string representation of the frames on the call stack at the time the current exception was thrown. |
| <a href="#">TargetSite</a> (inherited from Exception)     | Gets the method that throws the current exception.                                                         |

Public instance methods

| Member                                                                       | Description                                                                                                                                       |
|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">GetBaseException</a> (inherited from <a href="#">Exception</a> ) | When overridden in a derived class, returns the <a href="#">System.Exception</a> that is the root cause of one or more subsequent exceptions.     |
| <a href="#">GetObjectData</a> (inherited from <a href="#">Exception</a> )    | When overridden in a derived class, sets the <a href="#">System.Runtime.Serialization.SerializationInfo</a> with information about the exception. |
| <a href="#">ToString</a> (inherited from <a href="#">Exception</a> )         | Creates and returns a string representation of the current exception.                                                                             |

Protected instance properties

| Member                                                              | Description                                                                             |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#">HResult</a> (inherited from <a href="#">Exception</a> ) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

## QAException constructor

Create a QAException

Prototypes

```
' Visual Basic
Overloads Public Sub New( _
    ByVal msg As String _
)
```

```
// C#
public QAException(
    string msg
);
```

Parameters

◆ **msg** the exception description

## QAException constructor

Create a QAException

Prototypes

```
' Visual Basic
Overloads Public Sub New( _
    ByVal msg As String, _
    ByVal errCode As Integer _
)
```

---

```
// C#  
public QAEException(  
    string msg,  
    int errCode  
);
```

Parameters

- ◆ **msg** the exception description
- ◆ **errCode** the error code

## ErrorCode property

The error code of the exception

Prototypes

```
' Visual Basic  
Public Readonly Property ErrorCode As Integer  
  
// C#  
public int ErrorCode {get;}
```



## QAManager class

A manager for QA messaging operations. Unlike the transactional manager, message puts occur immediately and no other action is necessary to allow another client to get the message. Message gets are acknowledged via one of two modes. The implicit acknowledgement mode indicates that all messages are acknowledged as received as soon as the `getMessage` returns to the caller. Similarly, for message listeners, the message is acknowledged as soon as the call to the message listener delegate returns. The explicit acknowledgement mode indicates that messages are not acknowledged as received until a call to one of the acknowledge methods is made.

Prototypes

```

' Visual Basic
Public Class QAManager
    Inherits QAManagerBase

// C#
public class QAManager :
    QAManagerBase
  
```

### QAManager members

Public instance  
properties

| Member                                                                                 | Description                                                                                                                                       |
|----------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">LastError property</a> (inherited from <code>QAManagerBase</code> )        | The error code of the last executed method, with 0 indicating success.                                                                            |
| <a href="#">LastErrorMessage property</a> (inherited from <code>QAManagerBase</code> ) | The text of the error associated with the last executed method. Will be null if the last executed method result in a <code>LastError</code> of 0. |
| <a href="#">Mode property</a> (inherited from <code>QAManagerBase</code> )             | The acknowledgement mode for receiving all messages through this manager                                                                          |

Public instance methods

| Member                                  | Description                                                                                           |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------|
| <a href="#">Acknowledge method</a>      | Acknowledges the given message as received.                                                           |
| <a href="#">AcknowledgeAll method</a>   | Acknowledges all the unacknowledged messages as received.                                             |
| <a href="#">AcknowledgeUntil method</a> | Acknowledges all the unacknowledged messages received before and up to the given message as received. |

| Member                                                                        | Description                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BrowseMessages method</a> (inherited from QAManagerBase)          | Browse the next available messages waiting that have been sent to the given address. The messages are just being browsed, so they cannot be acknowledged. Enumerators returned from the same manager cannot have their method calls interlaced. Interlacing calls may result in messages meant for one iterator to be browsed in another iterator. |
| <a href="#">Close method</a> (inherited from QAManagerBase)                   | Closes the connection to the QA message system and releases any resources. Any calls to close beyond the first are ignored. Any subsequent calls to a method, other than close, will result in an exception being thrown.                                                                                                                          |
| <a href="#">CreateBinaryMessage method</a> (inherited from QAManagerBase)     | Create a BinaryMessage instance appropriate for sending.                                                                                                                                                                                                                                                                                           |
| <a href="#">CreateTextMessage method</a> (inherited from QAManagerBase)       | Create a TextMessage instance appropriate for sending.                                                                                                                                                                                                                                                                                             |
| <a href="#">GetBooleanStoreProperty method</a> (inherited from QAManagerBase) | Gets a boolean message store property                                                                                                                                                                                                                                                                                                              |
| <a href="#">GetDoubleStoreProperty method</a> (inherited from QAManagerBase)  | Gets a double message store property                                                                                                                                                                                                                                                                                                               |
| <a href="#">GetFloatStoreProperty method</a> (inherited from QAManagerBase)   | Gets a float message store property                                                                                                                                                                                                                                                                                                                |
| <a href="#">GetIntStoreProperty method</a> (inherited from QAManagerBase)     | Gets an int message store property                                                                                                                                                                                                                                                                                                                 |
| <a href="#">GetLongStoreProperty method</a> (inherited from QAManagerBase)    | Gets a long message store property                                                                                                                                                                                                                                                                                                                 |
| <a href="#">GetMessage method</a> (inherited from QAManagerBase)              | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call blocks indefinitely until a message is available.                                                                                                                                                                 |
| <a href="#">GetMessageNoWait method</a> (inherited from QAManagerBase)        | Get the next available message waiting that has been sent to the given address. If there is no message available, then it returns null immediately, without blocking.                                                                                                                                                                              |
| <a href="#">GetMessageTimeout method</a> (inherited from QAManagerBase)       | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call will wait up to the timeout time until a message is available.                                                                                                                                                    |
| <a href="#">GetSbyteStoreProperty method</a> (inherited from QAManagerBase)   | Gets a signed byte message store property                                                                                                                                                                                                                                                                                                          |

| Member                                                                           | Description                                                                                                                                                                                                                                                   |
|----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">GetShortStoreProperty method</a><br>(inherited from QAManagerBase)   | Gets a short message store property                                                                                                                                                                                                                           |
| <a href="#">GetStoreProperty method</a> (inherited from QAManagerBase)           | Gets a message store property                                                                                                                                                                                                                                 |
| <a href="#">GetStringStoreProperty method</a><br>(inherited from QAManagerBase)  | Gets a string message store property                                                                                                                                                                                                                          |
| <a href="#">Open method</a>                                                      | Open the manager with the give acknowledgement mode. The open method must be the first method called after creating a manager.                                                                                                                                |
| <a href="#">PutMessage method</a> (inherited from QAManagerBase)                 | Puts the message into the message system addressed to the given address.                                                                                                                                                                                      |
| <a href="#">PutMessageTimeToLive method</a><br>(inherited from QAManagerBase)    | Puts the message into the message system addressed to the given address, with the given time-to-live.                                                                                                                                                         |
| <a href="#">Recover method</a>                                                   | Force all unacknowledged messages into a state of unreceived. That is, these messages must be received again via <code>getMessage</code> .                                                                                                                    |
| <a href="#">SetBooleanStoreProperty method</a><br>(inherited from QAManagerBase) | Sets a boolean message store property                                                                                                                                                                                                                         |
| <a href="#">SetDoubleStoreProperty method</a><br>(inherited from QAManagerBase)  | Sets a double message store property                                                                                                                                                                                                                          |
| <a href="#">SetFloatStoreProperty method</a><br>(inherited from QAManagerBase)   | Sets a float message store property                                                                                                                                                                                                                           |
| <a href="#">SetIntStoreProperty method</a> (inherited from QAManagerBase)        | Sets an int message store property                                                                                                                                                                                                                            |
| <a href="#">SetLongStoreProperty method</a><br>(inherited from QAManagerBase)    | Sets a long message store property                                                                                                                                                                                                                            |
| <a href="#">SetMessageListener method</a> (inherited from QAManagerBase)         | Sets a listener for messages available for the given address. Only one listener can be set for a given address. Setting with a null listener clears out any listener for that address.                                                                        |
| <a href="#"> SetProperty method</a> (inherited from QAManagerBase)               | Sets the named property to the given value. Properties for this QAManagerBase may be set with this method as an alternative to the properties file at creation. Properties must be set before calling the <code>open()</code> methods of the derived classes. |
| <a href="#">SetSbyteStoreProperty method</a><br>(inherited from QAManagerBase)   | Sets a byte message store property                                                                                                                                                                                                                            |

| Member                                                                       | Description                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">SetShortStoreProperty method</a> (inherited from QAManagerBase)  | Sets a short message store property                                                                                                                                                                                                |
| <a href="#">SetStoreProperty method</a> (inherited from QAManagerBase)       | Sets a message store property. The property type must be one of the acceptable primitive types, or String.                                                                                                                         |
| <a href="#">SetStringStoreProperty method</a> (inherited from QAManagerBase) | Sets a string message store property                                                                                                                                                                                               |
| <a href="#">Start method</a> (inherited from QAManagerBase)                  | Once started the manager will receive any incoming messages. Any calls to start beyond the first without an intervening stop are ignored.                                                                                          |
| <a href="#">Stop method</a> (inherited from QAManagerBase)                   | Once stopped the manager will not receive any incoming messages. The messages are not lost. They just won't be received until the manager is started. Any calls to stop beyond the first without an intervening start are ignored. |
| <a href="#">TriggerSendReceive method</a> (inherited from QAManagerBase)     | Causes a synchronization with the QA message server, uploading any messages not meant for this client, and downloading any messages meant for this client.                                                                         |

Protected static fields  
(Shared)

| Member                                                       | Description                                    |
|--------------------------------------------------------------|------------------------------------------------|
| <a href="#">isOpen field</a> (inherited from QAManagerBase)  | Indicates whether instance is in an open state |
| <a href="#">mgrBase field</a> (inherited from QAManagerBase) | Handle to the underlying c++ qa manager        |

Protected instance fields

| Member                                                       | Description                                    |
|--------------------------------------------------------------|------------------------------------------------|
| <a href="#">isOpen field</a> (inherited from QAManagerBase)  | Indicates whether instance is in an open state |
| <a href="#">mgrBase field</a> (inherited from QAManagerBase) | Handle to the underlying c++ qa manager        |

Protected instance methods

| Member                         | Description                        |
|--------------------------------|------------------------------------|
| <a href="#">Dispose method</a> | Clean up any resources being used. |

## Acknowledge method

Acknowledges the given message as received.

Prototypes

```
' Visual Basic
Public Sub Acknowledge( _
    ByVal msg As QAMessage _
)
```

```
// C#
public void Acknowledge(
    QAMessage msg
);
```

Parameters

◆ **msg** the message to acknowledge

Exceptions

◆ [QAException class](#) - if there is a problem acknowledging the message

## AcknowledgeAll method

Acknowledges all the unacknowledged messages as received.

Prototypes

```
' Visual Basic
Public Sub AcknowledgeAll()
```

```
// C#
public void AcknowledgeAll();
```

Exceptions

◆ [QAException class](#) - if there is a problem acknowledging the messages

## AcknowledgeUntil method

Acknowledges all the unacknowledged messages received before and up to the given message as received.

Prototypes

```
' Visual Basic
Public Sub AcknowledgeUntil( _
    ByVal msg As QAMessage _
)
```

```
// C#
public void AcknowledgeUntil(
    QAMessage msg
);
```

Parameters

◆ **msg** the last message to acknowledge

---

Exceptions

- ◆ [QAEException class](#) - if there is a problem acknowledging the messages

## Dispose method

Clean up any resources being used.

Prototypes

```
' Visual Basic
Overloads Overrides Protected Sub Dispose( _
    ByVal disposing As Boolean _
)

// C#
protected override void Dispose(
    bool disposing
);
```

Parameters

- ◆ **disposing** true to release both managed and unmanaged resources; false to release only unmanaged resources.

## Open method

Open the manager with the give acknowledgement mode. The open method must be the first method called after creating a manager.

Prototypes

```
' Visual Basic
Public Sub Open( _
    ByVal mode As AcknowledgementMode _
)

// C#
public void Open(
    AcknowledgementMode mode
);
```

Parameters

- ◆ **mode** must be one of `AcknowledgementMode.EXPLICIT_ACKNOWLEDGEMENT` or `AcknowledgementMode.IMPLICIT_ACKNOWLEDGEMENT`

Exceptions

- ◆ [QAEException class](#) - if there is a problem opening the manager

## Recover method

Force all unacknowledged messages into a state of unreceived. That is, these messages must be received again via `getMessage`.

Prototypes

```
' Visual Basic
Public Sub Recover()

// C#
public void Recover();
```

Exceptions

- ◆ [QAEException class](#) - if there is a problem recovering

## QAManagerBase class

This class is an abstract base class for QA managers. It provides service for creating, sending, browsing and receiving messages. It is single threaded. That is, the thread that creates the manager is the only thread allowed to make calls into the manager.

Prototypes

```
' Visual Basic
Public Class QAManagerBase
    Inherits Component

// C#
public class QAManagerBase :
    Component
```

### QAManagerBase members

Public instance  
properties

| Member                                    | Description                                                                                                                          |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">LastError property</a>        | The error code of the last executed method, with 0 indicating success.                                                               |
| <a href="#">LastErrorMessage property</a> | The text of the error associated with the last executed method. Will be null if the last executed method result in a LastError of 0. |
| <a href="#">Mode property</a>             | The acknowledgement mode for receiving all messages through this manager                                                             |

Public instance methods

| Member                                     | Description                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BrowseMessages method</a>      | Browse the next available messages waiting that have been sent to the given address. The messages are just being browsed, so they cannot be acknowledged. Enumerators returned from the same manager cannot have their method calls interlaced. Interlacing calls may result in messages meant for one iterator to be browsed in another iterator. |
| <a href="#">Close method</a>               | Closes the connection to the QA message system and releases any resources. Any calls to close beyond the first are ignored. Any subsequent calls to a method, other than close, will result in an exception being thrown.                                                                                                                          |
| <a href="#">CreateBinaryMessage method</a> | Create a BinaryMessage instance appropriate for sending.                                                                                                                                                                                                                                                                                           |
| <a href="#">CreateTextMessage method</a>   | Create a TextMessage instance appropriate for sending.                                                                                                                                                                                                                                                                                             |

| <b>Member</b>                                  | <b>Description</b>                                                                                                                                                                              |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">GetBooleanStoreProperty method</a> | Gets a boolean message store property                                                                                                                                                           |
| <a href="#">GetDoubleStoreProperty method</a>  | Gets a double message store property                                                                                                                                                            |
| <a href="#">GetFloatStoreProperty method</a>   | Gets a float message store property                                                                                                                                                             |
| <a href="#">GetIntStoreProperty method</a>     | Gets an int message store property                                                                                                                                                              |
| <a href="#">GetLongStoreProperty method</a>    | Gets a long message store property                                                                                                                                                              |
| <a href="#">GetMessage method</a>              | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call blocks indefinitely until a message is available.              |
| <a href="#">GetMessageNoWait method</a>        | Get the next available message waiting that has been sent to the given address. If there is no message available, then it returns null immediately, without blocking.                           |
| <a href="#">GetMessageTimeout method</a>       | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call will wait up to the timeout time until a message is available. |
| <a href="#">GetSbyteStoreProperty method</a>   | Gets a signed byte message store property                                                                                                                                                       |
| <a href="#">GetShortStoreProperty method</a>   | Gets a short message store property                                                                                                                                                             |
| <a href="#">GetStoreProperty method</a>        | Gets a message store property                                                                                                                                                                   |
| <a href="#">GetStringStoreProperty method</a>  | Gets a string message store property                                                                                                                                                            |
| <a href="#">PutMessage method</a>              | Puts the message into the message system addressed to the given address.                                                                                                                        |
| <a href="#">PutMessageTimeToLive method</a>    | Puts the message into the message system addressed to the given address, with the given time-to-live.                                                                                           |
| <a href="#">SetBooleanStoreProperty method</a> | Sets a boolean message store property                                                                                                                                                           |
| <a href="#">SetDoubleStoreProperty method</a>  | Sets a double message store property                                                                                                                                                            |
| <a href="#">SetFloatStoreProperty method</a>   | Sets a float message store property                                                                                                                                                             |
| <a href="#">SetIntStoreProperty method</a>     | Sets an int message store property                                                                                                                                                              |
| <a href="#">SetLongStoreProperty method</a>    | Sets a long message store property                                                                                                                                                              |
| <a href="#">SetMessageListener method</a>      | Sets a listener for messages available for the given address. Only one listener can be set for a given address. Setting with a null listener clears out any listener for that address.          |



| Member                                         | Description                                                                                                                                                                                                                                      |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"> SetProperty method</a>            | Sets the named property to the given value. Properties for this QAManagerBase may be set with this method as an alternative to the properties file at creation. Properties must be set before calling the open() methods of the derived classes. |
| <a href="#"> SetByteStoreProperty method</a>   | Sets a byte message store property                                                                                                                                                                                                               |
| <a href="#"> SetShortStoreProperty method</a>  | Sets a short message store property                                                                                                                                                                                                              |
| <a href="#"> SetStoreProperty method</a>       | Sets a message store property. The property type must be one of the acceptable primitive types, or String.                                                                                                                                       |
| <a href="#"> SetStringStoreProperty method</a> | Sets a string message store property                                                                                                                                                                                                             |
| <a href="#"> Start method</a>                  | Once started the manager will receive any incoming messages. Any calls to start beyond the first without an intervening stop are ignored.                                                                                                        |
| <a href="#"> Stop method</a>                   | Once stopped the manager will not receive any incoming messages. The messages are not lost. They just won't be received until the manager is started. Any calls to stop beyond the first without an intervening start are ignored.               |
| <a href="#"> TriggerSendReceive method</a>     | Causes a synchronization with the QA message server, uploading any messages not meant for this client, and downloading any messages meant for this client.                                                                                       |

Protected static fields  
(Shared)

| Member                         | Description                                    |
|--------------------------------|------------------------------------------------|
| <a href="#"> isOpen field</a>  | Indicates whether instance is in an open state |
| <a href="#"> mgrBase field</a> | Handle to the underlying c++ qa manager        |

Protected instance fields

| Member                         | Description                                    |
|--------------------------------|------------------------------------------------|
| <a href="#"> isOpen field</a>  | Indicates whether instance is in an open state |
| <a href="#"> mgrBase field</a> | Handle to the underlying c++ qa manager        |

Protected instance  
methods

---

| Member                         | Description                        |
|--------------------------------|------------------------------------|
| <a href="#">Dispose method</a> | Clean up any resources being used. |

## isOpen field

Indicates whether instance is in an open state

Prototypes

**Visual Basic**  
 Family **isOpen** As Boolean

**C#**  
 family bool **isOpen**;

## mgrBase field

Handle to the underlying c++ qa manager

Prototypes

**Visual Basic**  
 Family **mgrBase** As IntPtr

**C#**  
 family IntPtr **mgrBase**;

## LastError property

The error code of the last executed method, with 0 indicating success.

Prototypes

**Visual Basic**  
 Public Readonly Property **LastError** As Integer

**C#**  
 public int **LastError** {get;}

## LastErrorMessage property

The text of the error associated with the last executed method. Will be null if the last executed method result in a LastError of 0.

Prototypes

**Visual Basic**  
 Public Readonly Property **LastErrorMessage** As String

**C#**  
 public string **LastErrorMessage** {get;}

## Mode property

The acknowledgement mode for receiving all messages through this manager

Prototypes

```
' Visual Basic
Public Readonly Property Mode As AcknowledgementMode

// C#
public AcknowledgementMode Mode {get;}
```

## BrowseMessages method

Browse the next available messages waiting that have been sent to the given address. The messages are just being browsed, so they cannot be acknowledged. Enumerators returned from the same manager cannot have their method calls interlaced. Interlacing calls may result in messages meant for one iterator to be browsed in another iterator.

Prototypes

```
' Visual Basic
Public Function BrowseMessages( _
    ByVal address As String _
) As System.Collections.IEnumerator

// C#
public System.Collections.IEnumerator BrowseMessages(
    string address
);
```

Parameters

◆ **address** the address of the messages

Return value

an enumerator over the available messages

## Close method

Closes the connection to the QA message system and releases any resources. Any calls to close beyond the first are ignored. Any subsequent calls to a method, other than close, will result in an exception being thrown.

Prototypes

```
' Visual Basic
Public Sub Close()

// C#
public void Close();
```

Exceptions

◆ [QAException class](#) - if there is a problem closing the manager.

---

## CreateBinaryMessage method

Create a BinaryMessage instance appropriate for sending.

Prototypes

```
' Visual Basic
Public Function CreateBinaryMessage() As QABinaryMessage

// C#
public QABinaryMessage CreateBinaryMessage();
```

Return value

a new BinaryMessage

Exceptions

◆ [QAEException class](#) - if there is a problem creating the message.

## CreateTextMessage method

Create a TextMessage instance appropriate for sending.

Prototypes

```
' Visual Basic
Public Function CreateTextMessage() As QATextMessage

// C#
public QATextMessage CreateTextMessage();
```

Return value

a new TextMessage

Exceptions

◆ [QAEException class](#) - if there is a problem creating the message.

## Dispose method

Clean up any resources being used.

Prototypes

```
' Visual Basic
Overloads Overrides Protected Sub Dispose( _
    ByVal disposing As Boolean _
)

// C#
protected override void Dispose(
    bool disposing
);
```

Parameters

◆ **disposing** true to release both managed and unmanaged resources;  
false to release only unmanaged resources.

## GetBooleanStoreProperty method

Gets a boolean message store property

|              |                                                                                                                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>' Visual Basic</b> Public Function <b>GetBooleanStoreProperty</b>( _     ByVal <i>propName</i> As String _ ) As Boolean  <b>// C#</b> public bool <b>GetBooleanStoreProperty</b>(     string <i>propName</i> ); </pre> |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                                             |
| Return value | the property value                                                                                                                                                                                                              |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                                               |

## GetDoubleStoreProperty method

Gets a double message store property

|              |                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>' Visual Basic</b> Public Function <b>GetDoubleStoreProperty</b>( _     ByVal <i>propName</i> As String _ ) As Double  <b>// C#</b> public double <b>GetDoubleStoreProperty</b>(     string <i>propName</i> ); </pre> |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                                            |
| Return value | the property value                                                                                                                                                                                                             |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                                              |

## GetFloatStoreProperty method

Gets a float message store property

|            |                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>GetFloatStoreProperty</b>( _     ByVal <i>propName</i> As String _ ) As Single  <b>// C#</b> public float <b>GetFloatStoreProperty</b>(     string <i>propName</i> ); </pre> |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                               |
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetIntStoreProperty method

Gets an int message store property

|            |                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <b>' Visual Basic</b><br>Public Function <b>GetIntStoreProperty</b> ( _<br>ByVal <i>propName</i> As String _<br>) As Integer<br><br><b>// C#</b><br>public int <b>GetIntStoreProperty</b> (<br>string <i>propName</i><br>); |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                               |
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetLongStoreProperty method

Gets a long message store property

|            |                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <b>' Visual Basic</b><br>Public Function <b>GetLongStoreProperty</b> ( _<br>ByVal <i>propName</i> As String _<br>) As Long<br><br><b>// C#</b><br>public long <b>GetLongStoreProperty</b> (<br>string <i>propName</i><br>); |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                               |
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetMessage method

Get the next available message waiting that has been sent to the given address. If there is no message available, then this call blocks indefinitely until a message is available.

Prototypes

```
' Visual Basic  
Public Function GetMessage( _  
    ByVal address As String _  
) As QAMessage
```

```
// C#  
public QAMessage GetMessage(  
    string address  
);
```

Parameters

◆ **address** the address of the message

Return value

the next available message

Exceptions

◆ [QAException class](#) - if there is a problem getting the message.

## GetMessageNoWait method

Get the next available message waiting that has been sent to the given address. If there is no message available, then it returns null immediately, without blocking.

Prototypes

```
' Visual Basic  
Public Function GetMessageNoWait( _  
    ByVal address As String _  
) As QAMessage
```

```
// C#  
public QAMessage GetMessageNoWait(  
    string address  
);
```

Parameters

◆ **address** the address of the message

Return value

the next available message or null there is no available message

Exceptions

◆ [QAException class](#) - if there is a problem getting the message.

## GetMessageTimeout method

Get the next available message waiting that has been sent to the given address. If there is no message available, then this call will wait up to the timeout time until a message is available.

---

|              |                                                                                                                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>' Visual Basic</b> Public Function <b>GetMessageTimeout</b>( _     ByVal <i>address</i> As String, _     ByVal <i>timeout</i> As Long _ ) As QAMessage  <b>// C#</b> public QAMessage <b>GetMessageTimeout</b>(     string <i>address</i>,     long <i>timeout</i> ); </pre> |
| Parameters   | <ul style="list-style-type: none"> <li>◆ <b>address</b> the address of the message</li> <li>◆ <b>timeout</b> the time to wait, in milliseconds, for a message to become available</li> </ul>                                                                                          |
| Return value | the next available message or null there is no available message                                                                                                                                                                                                                      |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if there is a problem getting the message.                                                                                                                                                                                                     |

## GetSbyteStoreProperty method

Gets a signed byte message store property

|              |                                                                                                                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>' Visual Basic</b> Public Function <b>GetSbyteStoreProperty</b>( _     ByVal <i>propName</i> As String _ ) As System.SByte  <b>// C#</b> public System.Sbyte <b>GetSbyteStoreProperty</b>(     string <i>propName</i> ); </pre> |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                                                      |
| Return value | the property value                                                                                                                                                                                                                       |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                                                       |

## GetShortStoreProperty method

Gets a short message store property

|            |                                                                                                                                     |
|------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>GetShortStoreProperty</b>( _     ByVal <i>propName</i> As String _ ) As Short </pre> |
|------------|-------------------------------------------------------------------------------------------------------------------------------------|



```
// C#
public short GetShortStoreProperty(
    string propName
);
```

|              |                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                                |
| Return value | the property value                                                                                                                 |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetStoreProperty method

Gets a message store property

```
Prototypes
' Visual Basic
Public Function GetStoreProperty( _
    ByVal propName As String _
) As Object
```

```
// C#
public object GetStoreProperty(
    string propName
);
```

|              |                                                                       |
|--------------|-----------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                   |
| Return value | the property value                                                    |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if the property does not exist |

## GetStringStoreProperty method

Gets a string message store property

```
Prototypes
' Visual Basic
Public Function GetStringStoreProperty( _
    ByVal propName As String _
) As String
```

```
// C#
public string GetStringStoreProperty(
    string propName
);
```

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                       |
| Return value | the property value or null if the property does not exist |

---

## PutMessage method

Puts the message into the message system addressed to the given address.

Prototypes

```
' Visual Basic
Public Sub PutMessage( _
    ByVal address As String, _
    ByVal msg As QAMessage _
)

// C#
public void PutMessage(
    string address,
    QAMessage msg
);
```

Parameters

- ◆ **address** the address of the message
- ◆ **msg** the message to put

Exceptions

- ◆ [QAException class](#) - if there is a problem putting the message.

## PutMessageTimeToLive method

Puts the message into the message system addressed to the given address, with the given time-to-live.

Prototypes

```
' Visual Basic
Public Sub PutMessageTimeToLive( _
    ByVal address As String, _
    ByVal msg As QAMessage, _
    ByVal ttl As Long _
)

// C#
public void PutMessageTimeToLive(
    string address,
    QAMessage msg,
    long ttl
);
```

Parameters

- ◆ **address** the address of the message
- ◆ **msg** the message to put
- ◆ **ttl** the delay, in milliseconds, before the message will expire if it has not been delivered. A value of 0 indicates the message will not expire.

Exceptions

- ◆ [QAException class](#) - if there is a problem putting the message.

## SetBooleanStoreProperty method

Sets a boolean message store property

Prototypes

```

' Visual Basic
Public Sub SetBooleanStoreProperty( _
    ByVal propName As String, _
    ByVal val As Boolean _
)

```

```

// C#
public void SetBooleanStoreProperty(
    string propName,
    bool val
);

```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetDoubleStoreProperty method

Sets a double message store property

Prototypes

```

' Visual Basic
Public Sub SetDoubleStoreProperty( _
    ByVal propName As String, _
    ByVal val As Double _
)

```

```

// C#
public void SetDoubleStoreProperty(
    string propName,
    double val
);

```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetFloatStoreProperty method

Sets a float message store property

Prototypes

```

' Visual Basic
Public Sub SetFloatStoreProperty( _
    ByVal propName As String, _
    ByVal val As Single _
)

```

---

```
// C#  
public void SetFloatStoreProperty(  
    string propName,  
    float val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetIntStoreProperty method

Sets an int message store property

Prototypes

```
' Visual Basic  
Public Sub SetIntStoreProperty( _  
    ByVal propName As String, _  
    ByVal val As Integer _  
)
```

```
// C#  
public void SetIntStoreProperty(  
    string propName,  
    int val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetLongStoreProperty method

Sets a long message store property

Prototypes

```
' Visual Basic  
Public Sub SetLongStoreProperty( _  
    ByVal propName As String, _  
    ByVal val As Long _  
)
```

```
// C#  
public void SetLongStoreProperty(  
    string propName,  
    long val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetMessageListener method

Sets a listener for messages available for the given address. Only one listener can be set for a given address. Setting with a null listener clears out any listener for that address.

Prototypes

```
' Visual Basic
Public Sub SetMessageListener( _
    ByVal address As String, _
    ByVal listener As QAManagerBase.MessageListener _
)

// C#
public void SetMessageListener(
    string address,
    QAManagerBase.MessageListener listener
);
```

Parameters

- ◆ **address** the address of messages
- ◆ **listener** the listener

## SetProperty method

Sets the named property to the given value. Properties for this QAManagerBase may be set with this method as an alternative to the properties file at creation. Properties must be set before calling the open() methods of the derived classes.

Prototypes

```
' Visual Basic
Public Sub SetProperty( _
    ByVal name As String, _
    ByVal val As String _
)

// C#
public void SetProperty(
    string name,
    string val
);
```

Parameters

- ◆ **name** the property name
- ◆ **val** the property value

Exceptions

- ◆ [QAException class](#) - if there is a problem setting the property.

## SetSbyteStoreProperty method

Sets a byte message store property

---

|            |                                                                                                                                                                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Sub <b>SetSbyteStoreProperty</b>( _     ByVal <i>propName</i> As String, _     ByVal <i>val</i> As System.SByte _ )  <b>// C#</b> public void <b>SetSbyteStoreProperty</b>(     string <i>propName</i>,     System.Sbyte <i>val</i> ); </pre> |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>propName</b> the property name</li> <li>◆ <b>val</b> the property value</li> </ul>                                                                                                                                                   |

## SetShortStoreProperty method

Sets a short message store property

|            |                                                                                                                                                                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Sub <b>SetShortStoreProperty</b>( _     ByVal <i>propName</i> As String, _     ByVal <i>val</i> As Short _ )  <b>// C#</b> public void <b>SetShortStoreProperty</b>(     string <i>propName</i>,     short <i>val</i> ); </pre> |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>propName</b> the property name</li> <li>◆ <b>val</b> the property value</li> </ul>                                                                                                                                     |

## SetStoreProperty method

Sets a message store property. The property type must be one of the acceptable primitive types, or String.

|            |                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Sub <b>SetStoreProperty</b>( _     ByVal <i>propName</i> As String, _     ByVal <i>val</i> As Object _ ) </pre> |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------|

```
// C#
public void SetStoreProperty(
    string propName,
    object val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetStringStoreProperty method

Sets a string message store property

Prototypes

```
' Visual Basic
Public Sub SetStringStoreProperty( _
    ByVal propName As String, _
    ByVal val As String _
)
```

```
// C#
public void SetStringStoreProperty(
    string propName,
    string val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## Start method

Once started the manager will receive any incoming messages. Any calls to start beyond the first without an intervening stop are ignored.

Prototypes

```
' Visual Basic
Public Sub Start()
```

```
// C#
public void Start();
```

Exceptions

- ◆ [QAException class](#) - if there is a problem starting the manager.

## Stop method

Once stopped the manager will not receive any incoming messages. The messages are not lost. They just won't be received until the manager is started. Any calls to stop beyond the first without an intervening start are ignored.

---

Prototypes

' **Visual Basic**  
Public Sub **Stop()**

// **C#**  
public void **Stop();**

Exceptions

- ◆ [QAException class](#) - if there is a problem stopping the manager.

## TriggerSendReceive method

Causes a synchronization with the QA message server, uploading any messages not meant for this client, and downloading any messages meant for this client.

Prototypes

' **Visual Basic**  
Public Sub **TriggerSendReceive()**

// **C#**  
public void **TriggerSendReceive();**

Exceptions

- ◆ [QAException class](#) - if there is a problem triggering the send/receive.



## QAManagerBase.MessageListener delegate

MessageListener delegate definition

Prototypes

```
' Visual Basic  
Delegate Sub QAManagerBase.MessageListener( _  
    ByVal msg As QAMessage _  
)
```

```
// C#  
delegate void QAManagerBase.MessageListener(  
    QAMessage msg  
);
```

Parameters

◆ **msg** the message that was received

---

# QAManagerFactory class

Factory for creating QAManager and QATransactionalManager objects.  
There is only ever one instance of QAManagerFactory,

Prototypes

**Visual Basic**  
Public Class **QAManagerFactory**  
Inherits Component

**C#**  
public class **QAManagerFactory** :  
Component

## QAManagerFactory members

Public static fields  
(Shared)

| Member                           | Description |
|----------------------------------|-------------|
| <a href="#">InstanceID field</a> | Factory id  |

Public static properties  
(Shared)

| Member                                 | Description                               |
|----------------------------------------|-------------------------------------------|
| <a href="#">Instance property</a>      | A singleton QAManagerFactory instance.    |
| <a href="#">InstanceCount property</a> | Indicates the number of factory instances |

Public instance fields

| Member                           | Description |
|----------------------------------|-------------|
| <a href="#">InstanceID field</a> | Factory id  |

Public instance  
properties

| Member                                    | Description                                                                                                                          |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">LastError property</a>        | The error code of the last executed method, with 0 indicating success.                                                               |
| <a href="#">LastErrorMessage property</a> | The text of the error associated with the last executed method. Will be null if the last executed method result in a LastError of 0. |

## Public instance methods

| Member                                              | Description                                                     |
|-----------------------------------------------------|-----------------------------------------------------------------|
| <a href="#">CreateQAManager method</a>              | Create a QA Manager configured from the INI file.               |
| <a href="#">CreateQATransactionalManager method</a> | Create a transactional QA Manager configured from the INI file. |

## Protected instance methods

| Member                          | Description                        |
|---------------------------------|------------------------------------|
| <a href="#">Dispose method</a>  | Clean up any resources being used. |
| <a href="#">Finalize method</a> | Cleanup the instance               |

**InstanceID field**

Factory id

## Prototypes

```

Visual Basic
Public InstanceID As Integer

C#
public int InstanceID;

```

**Instance property**

A singleton QAManagerFactory instance.

## Prototypes

```

Visual Basic
Public Shared ReadOnly Property Instance As QAManagerFactory

C#
public const QAManagerFactory Instance {get;}

```

## Exceptions

- ◆ [QAEException class](#) - when there is a problem creating the manager factory

**InstanceCount property**

Indicates the number of factory instances

## Prototypes

```

Visual Basic
Public Shared ReadOnly Property InstanceCount As Long

```

---

```
// C#
public const long InstanceCount {get;}
```

## LastError property

The error code of the last executed method, with 0 indicating success.

Prototypes

```
· Visual Basic
Public Readonly Property LastError As Integer
```

```
// C#
public int LastError {get;}
```

## LastErrorMessage property

The text of the error associated with the last executed method. Will be null if the last executed method result in a LastError of 0.

Prototypes

```
· Visual Basic
Public Readonly Property LastErrorMessage As String
```

```
// C#
public string LastErrorMessage {get;}
```

## CreateQAManager method

Create a QA Manager configured from the INI file.

Prototypes

```
· Visual Basic
Public Function CreateQAManager( _
    ByVal iniFile As String _
) As QAManager
```

```
// C#
public QAManager CreateQAManager(
    string iniFile
);
```

Parameters

◆ **iniFile** properties file configuring the QAManager instance

Return value

the configured QAManager

Exceptions

◆ [QAException class](#) - when there is a problem creating the manager

## CreateQATransactionalManager method

Create a transactional QA Manager configured from the INI file.

|              |                                                                                                                                                                                                                                                                          |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> <b>' Visual Basic</b> Public Function <b>CreateQATransactionalManager</b>( _     ByVal <i>iniFile</i> As String _ ) As QATransactionalManager  <b>// C#</b> public QATransactionalManager <b>CreateQATransactionalManager</b>(     string <i>iniFile</i> ); </pre> |
| Parameters   | <ul style="list-style-type: none"> <li>◆ <b>iniFile</b> properties file configuring the QATransactionalManager instance</li> </ul>                                                                                                                                       |
| Return value | the configured QATransactionalManager                                                                                                                                                                                                                                    |
| Exceptions   | <ul style="list-style-type: none"> <li>◆ <a href="#">QAException class</a> - when there is a problem creating the manager</li> </ul>                                                                                                                                     |

## Dispose method

Clean up any resources being used.

|            |                                                                                                                                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Overloads Overrides Protected Sub <b>Dispose</b>( _     ByVal <i>disposing</i> As Boolean _ )  <b>// C#</b> protected override void <b>Dispose</b>(     bool <i>disposing</i> ); </pre> |
| Parameters | <ul style="list-style-type: none"> <li>◆ <b>disposing</b> true to release both managed and unmanaged resources; false to release only unmanaged resources.</li> </ul>                                               |

## Finalize method

Cleanup the instance

|            |                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Overrides Protected Sub <b>Finalize</b>()  <b>// C#</b> protected override void <b>Finalize</b>(); </pre> |
|------------|---------------------------------------------------------------------------------------------------------------------------------------|

---

# QAMessage class

Encapsulates a QA message

Prototypes

**Visual Basic**  
Public Class **QAMessage**  
Inherits Component

**C#**  
public class **QAMessage** :  
Component

## QAMessage members

Public instance  
properties

| Member                                  | Description                                                                                                                  |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Address property</a>        | The address of this message. May be null, but is null never null in a message received via getMessage or a message listener. |
| <a href="#">Expiration property</a>     | Indicates the time after which the message may expire and be removed from the message system if it has not been received.    |
| <a href="#">InReplyToID property</a>    | The message id of the message for which this message is a reply. May be null.                                                |
| <a href="#">MessageID property</a>      | The globally unique message id of the message. This property is null until a message is put.                                 |
| <a href="#">Priority property</a>       | The priority of the message (ranging from 0 to 9)                                                                            |
| <a href="#">Redelivered property</a>    | Indicates whether the message has been previously received but not acknowledged.                                             |
| <a href="#">ReplyToAddress property</a> | The replyTo address of this message. May be null.                                                                            |
| <a href="#">Timestamp property</a>      | The message timestamp.                                                                                                       |

Public instance methods

| Member                                    | Description                             |
|-------------------------------------------|-----------------------------------------|
| <a href="#">ClearProperties method</a>    | Clear all the properties of the message |
| <a href="#">GetBooleanProperty method</a> | Gets a boolean message property         |
| <a href="#">GetDoubleProperty method</a>  | Gets a double message property          |
| <a href="#">GetFloatProperty method</a>   | Gets a float message property           |

| Member                                    | Description                                                                                  |
|-------------------------------------------|----------------------------------------------------------------------------------------------|
| <a href="#">GetIntProperty method</a>     | Gets an int message property                                                                 |
| <a href="#">GetLongProperty method</a>    | Gets a long message property                                                                 |
| <a href="#">GetProperty method</a>        | Gets a message property                                                                      |
| <a href="#">GetPropertyNames method</a>   | Gets an enumerator over the property names of the message                                    |
| <a href="#">GetPropertyType method</a>    | Returns the property type of the given property                                              |
| <a href="#">GetSbyteProperty method</a>   | Gets a signed byte message property                                                          |
| <a href="#">GetShortProperty method</a>   | Gets a short message property                                                                |
| <a href="#">GetStringProperty method</a>  | Gets a string message property                                                               |
| <a href="#">PropertyExists method</a>     | Indicates whether the given property has been set for this message                           |
| <a href="#">SetBooleanProperty method</a> | Sets a boolean property                                                                      |
| <a href="#">SetDoubleProperty method</a>  | Sets a double property                                                                       |
| <a href="#">SetFloatProperty method</a>   | Sets a float property                                                                        |
| <a href="#">SetIntProperty method</a>     | Sets an int property                                                                         |
| <a href="#">SetLongProperty method</a>    | Sets a long property                                                                         |
| <a href="#">SetProperty method</a>        | Sets a property. The property type must be one of the acceptable primitive types, or String. |
| <a href="#">SetSbyteProperty method</a>   | Sets a byte property                                                                         |
| <a href="#">SetShortProperty method</a>   | Sets a short property                                                                        |
| <a href="#">SetStringProperty method</a>  | Sets a string property                                                                       |

Protected instance  
methods

| Member                         | Description                        |
|--------------------------------|------------------------------------|
| <a href="#">Dispose method</a> | Clean up any resources being used. |

## Address property

The address of this message. May be null, but is null never null in a message received via `getMessage` or a message listener.

Prototypes

**Visual Basic**  
Public Property **Address** As String

---

```
// C#  
public string Address {get;set;}
```

## Expiration property

Indicates the time after which the message may expire and be removed from the message system if it has not been received.

Prototypes

```
· Visual Basic  
Public Readonly Property Expiration As Long
```

```
// C#  
public long Expiration {get;}
```

## InReplyToID property

The message id of the message for which this message is a reply. May be null.

Prototypes

```
· Visual Basic  
Public Property InReplyToID As String
```

```
// C#  
public string InReplyToID {get;set;}
```

## MessageID property

The globally unique message id of the message. This property is null until a message is put.

Prototypes

```
· Visual Basic  
Public Readonly Property MessageID As String
```

```
// C#  
public string MessageID {get;}
```

## Priority property

The priority of the message (ranging from 0 to 9)

Prototypes

```
· Visual Basic  
Public Property Priority As Integer
```

```
// C#  
public int Priority {get;set;}
```



## Redelivered property

Indicates whether the message has been previously received but not acknowledged.

Prototypes

```
' Visual Basic
Public Readonly Property Redelivered As Boolean

// C#
public bool Redelivered {get;}
```

## ReplyToAddress property

The replyTo address of this message. May be null.

Prototypes

```
' Visual Basic
Public Property ReplyToAddress As String

// C#
public string ReplyToAddress {get;set;}
```

## Timestamp property

The message timestamp.

Prototypes

```
' Visual Basic
Public Readonly Property Timestamp As Date

// C#
public DateTime Timestamp {get;}
```

## ClearProperties method

Clear all the properties of the message

Prototypes

```
' Visual Basic
Public Sub ClearProperties()

// C#
public void ClearProperties();
```

## Dispose method

Clean up any resources being used.

---

|            |                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <b>Visual Basic</b><br>Overloads Overrides Protected Sub <b>Dispose</b> (<br>ByVal <i>disposing</i> As Boolean _<br>)<br><br><b>C#</b><br>protected override void <b>Dispose</b> (<br>bool <i>disposing</i><br>); |
| Parameters | ♦ <b>disposing</b> true to release both managed and unmanaged resources;<br>false to release only unmanaged resources.                                                                                            |

## GetBooleanProperty method

Gets a boolean message property

|              |                                                                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <b>Visual Basic</b><br>Public Function <b>GetBooleanProperty</b> (<br>ByVal <i>propName</i> As String _<br>) As Boolean<br><br><b>C#</b><br>public bool <b>GetBooleanProperty</b> (<br>string <i>propName</i><br>); |
| Parameters   | ♦ <b>propName</b> the property name                                                                                                                                                                                 |
| Return value | the property value                                                                                                                                                                                                  |
| Exceptions   | ♦ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                                   |

## GetDoubleProperty method

Gets a double message property

|            |                                                                                                                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <b>Visual Basic</b><br>Public Function <b>GetDoubleProperty</b> (<br>ByVal <i>propName</i> As String _<br>) As Double<br><br><b>C#</b><br>public double <b>GetDoubleProperty</b> (<br>string <i>propName</i><br>); |
| Parameters | ♦ <b>propName</b> the property name                                                                                                                                                                                |

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetFloatProperty method

Gets a float message property

|            |                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>GetFloatProperty</b>( _     ByVal <i>propName</i> As String _ ) As Single  <b>// C#</b> public float <b>GetFloatProperty</b>(     string <i>propName</i> ); </pre> |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                               |
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetIntProperty method

Gets an int message property

|            |                                                                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>GetIntProperty</b>( _     ByVal <i>propName</i> As String _ ) As Integer  <b>// C#</b> public int <b>GetIntProperty</b>(     string <i>propName</i> ); </pre> |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|              |                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Parameters   | ◆ <b>propName</b> the property name                                                                                               |
| Return value | the property value                                                                                                                |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetLongProperty method

Gets a long message property

---

|              |                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <b>Visual Basic</b><br>Public Function <b>GetLongProperty</b> ( _<br>ByVal <i>propName</i> As String _<br>) As Long<br><br><b>C#</b><br>public long <b>GetLongProperty</b> (<br>string <i>propName</i><br>); |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                          |
| Return value | the property value                                                                                                                                                                                           |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                           |

## GetProperty method

Gets a message property

|              |                                                                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <b>Visual Basic</b><br>Public Function <b>GetProperty</b> ( _<br>ByVal <i>propName</i> As String _<br>) As Object<br><br><b>C#</b><br>public object <b>GetProperty</b> (<br>string <i>propName</i><br>); |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                      |
| Return value | the property value                                                                                                                                                                                       |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if the property does not exist                                                                                                                                    |

## GetPropertyNames method

Gets an enumerator over the property names of the message

|              |                                                                                                                                                                                           |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <b>Visual Basic</b><br>Public Function <b>GetPropertyNames</b> () As System.Collections.IEnumerator<br><br><b>C#</b><br>public System.Collections.IEnumerator <b>GetPropertyNames</b> (); |
| Return value | an enumerator over the message property names                                                                                                                                             |

## GetPropertyType method

Returns the property type of the given property

|              |                                                                                                                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> ' Visual Basic Public Function <b>GetPropertyType</b>( _     ByVal <i>propName</i> As String _ ) As QAPropertyType  // C# public QAPropertyType <b>GetPropertyType</b>(     string <i>propName</i> ); </pre> |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                                |
| Return value | the property type                                                                                                                                                                                                  |

## GetSbyteProperty method

Gets a signed byte message property

|              |                                                                                                                                                                                                                  |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes   | <pre> ' Visual Basic Public Function <b>GetSbyteProperty</b>( _     ByVal <i>propName</i> As String _ ) As System.SByte  // C# public System.Sbyte <b>GetSbyteProperty</b>(     string <i>propName</i> ); </pre> |
| Parameters   | ◆ <b>propName</b> the property name                                                                                                                                                                              |
| Return value | the property value                                                                                                                                                                                               |
| Exceptions   | ◆ <a href="#">QAException class</a> - if there is a conversion error getting the property value or if the property does not exist                                                                                |

## GetShortProperty method

Gets a short message property

|            |                                                                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> ' Visual Basic Public Function <b>GetShortProperty</b>( _     ByVal <i>propName</i> As String _ ) As Short  // C# public short <b>GetShortProperty</b>(     string <i>propName</i> ); </pre> |
| Parameters | ◆ <b>propName</b> the property name                                                                                                                                                                |

---

|              |                                                                                                                                    |
|--------------|------------------------------------------------------------------------------------------------------------------------------------|
| Return value | the property value                                                                                                                 |
| Exceptions   | ◆ <a href="#">QAEException class</a> - if there is a conversion error getting the property value or if the property does not exist |

## GetStringProperty method

Gets a string message property

|            |                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>GetStringProperty</b>( _     ByVal <i>propName</i> As String _ ) As String  <b>// C#</b> public string <b>GetStringProperty</b>(     string <i>propName</i> ); </pre> |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|            |                                     |
|------------|-------------------------------------|
| Parameters | ◆ <b>propName</b> the property name |
|------------|-------------------------------------|

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Return value | the property value or null if the property does not exist |
|--------------|-----------------------------------------------------------|

## PropertyExists method

Indicates whether the given property has been set for this message

|            |                                                                                                                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Function <b>PropertyExists</b>( _     ByVal <i>propName</i> As String _ ) As Boolean  <b>// C#</b> public bool <b>PropertyExists</b>(     string <i>propName</i> ); </pre> |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|            |                                     |
|------------|-------------------------------------|
| Parameters | ◆ <b>propName</b> the property name |
|------------|-------------------------------------|

|              |                             |
|--------------|-----------------------------|
| Return value | whether the property exists |
|--------------|-----------------------------|

## SetBooleanProperty method

Sets a boolean property

|            |                                                                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Prototypes | <pre> <b>' Visual Basic</b> Public Sub <b>SetBooleanProperty</b>( _     ByVal <i>propName</i> As String, _     ByVal <i>val</i> As Boolean _ ) </pre> |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

```
// C#
public void SetBooleanProperty(
    string propName,
    bool val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetDoubleProperty method

Sets a double property

Prototypes

```
' Visual Basic
Public Sub SetDoubleProperty( _
    ByVal propName As String, _
    ByVal val As Double _
)
```

```
// C#
public void SetDoubleProperty(
    string propName,
    double val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetFloatProperty method

Sets a float property

Prototypes

```
' Visual Basic
Public Sub SetFloatProperty( _
    ByVal propName As String, _
    ByVal val As Single _
)
```

```
// C#
public void SetFloatProperty(
    string propName,
    float val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

---

## SetIntProperty method

Sets an int property

Prototypes

```
' Visual Basic
Public Sub SetIntProperty( _
    ByVal propName As String, _
    ByVal val As Integer _
)

// C#
public void SetIntProperty(
    string propName,
    int val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetLongProperty method

Sets a long property

Prototypes

```
' Visual Basic
Public Sub SetLongProperty( _
    ByVal propName As String, _
    ByVal val As Long _
)

// C#
public void SetLongProperty(
    string propName,
    long val
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetProperty method

Sets a property. The property type must be one of the acceptable primitive types, or String.

Prototypes

```
' Visual Basic
Public Sub SetProperty( _
    ByVal propName As String, _
    ByVal val As Object _
)
```



```
// C#  
public void SetProperty(  
    string propName,  
    object val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetSbyteProperty method

Sets a byte property

Prototypes

```
' Visual Basic  
Public Sub SetSbyteProperty( _  
    ByVal propName As String, _  
    ByVal val As System.SByte _  
)
```

```
// C#  
public void SetSbyteProperty(  
    string propName,  
    System.Sbyte val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## SetShortProperty method

Sets a short property

Prototypes

```
' Visual Basic  
Public Sub SetShortProperty( _  
    ByVal propName As String, _  
    ByVal val As Short _  
)
```

```
// C#  
public void SetShortProperty(  
    string propName,  
    short val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

---

## SetStringProperty method

Sets a string property

Prototypes

```
' Visual Basic  
Public Sub SetStringProperty( _  
    ByVal propName As String, _  
    ByVal val As String _  
)
```

```
// C#  
public void SetStringProperty(  
    string propName,  
    string val  
);
```

Parameters

- ◆ **propName** the property name
- ◆ **val** the property value

## QAPROPERTYTYPE enumeration

QAMessage property type enumeration, corresponding naturally to the C# types

### Prototypes

```
· Visual Basic  
Public Enum QAPROPERTYTYPE  
  
// C#  
public enum QAPROPERTYTYPE
```

### Members

| Member  | Description                                                                 |
|---------|-----------------------------------------------------------------------------|
| BOOLEAN | Indicates a boolean property                                                |
| BYTE    | Indicates a signed byte property                                            |
| DOUBLE  | Indicates a double property                                                 |
| FLOAT   | Indicates a float property                                                  |
| INT     | Indicates an int property                                                   |
| LONG    | Indicates an long property                                                  |
| SHORT   | Indicates a short property                                                  |
| STRING  | Indicates a string property                                                 |
| UNKNOWN | Indicates an unknown property type, usually because the property is unknown |

---

# QATextMessage class

Encapsulation of a text message.

Prototypes

**Visual Basic**  
Public Class **QATextMessage**  
Inherits QAMessage

**C#**  
public class **QATextMessage** :  
QAMessage

## QATextMessage members

Public instance  
properties

| Member                                                             | Description                                                                                                                                                                            |
|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Address property</a> (inherited from QAMessage)        | The address of this message. May be null, but is null never null in a message received via getMessage or a message listener.                                                           |
| <a href="#">Expiration property</a> (inherited from QAMessage)     | Indicates the time after which the message may expire and be removed from the message system if it has not been received.                                                              |
| <a href="#">InReplyToID property</a> (inherited from QAMessage)    | The message id of the message for which this message is a reply. May be null.                                                                                                          |
| <a href="#">MessageID property</a> (inherited from QAMessage)      | The globally unique message id of the message. This property is null until a message is put.                                                                                           |
| <a href="#">Priority property</a> (inherited from QAMessage)       | The priority of the message (ranging from 0 to 9)                                                                                                                                      |
| <a href="#">Redelivered property</a> (inherited from QAMessage)    | Indicates whether the message has been previously received but not acknowledged.                                                                                                       |
| <a href="#">ReplyToAddress property</a> (inherited from QAMessage) | The replyTo address of this message. May be null.                                                                                                                                      |
| <a href="#">Text property</a>                                      | The message text. If the message exceeds the maximum message chunk size, then the Text value will be null. In the latter case, you should use the readText method to receive the text. |
| <a href="#">TextLength property</a>                                | The length, in characters, of the message                                                                                                                                              |
| <a href="#">Timestamp property</a> (inherited from QAMessage)      | The message timestamp.                                                                                                                                                                 |

Public instance methods

| Member                                                                             | Description                                                                                                                                                             |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ClearProperties method</a> (inherited from <code>QAMessage</code> )    | Clear all the properties of the message                                                                                                                                 |
| <a href="#">GetBooleanProperty method</a> (inherited from <code>QAMessage</code> ) | Gets a boolean message property                                                                                                                                         |
| <a href="#">GetDoubleProperty method</a> (inherited from <code>QAMessage</code> )  | Gets a double message property                                                                                                                                          |
| <a href="#">GetFloatProperty method</a> (inherited from <code>QAMessage</code> )   | Gets a float message property                                                                                                                                           |
| <a href="#">GetIntProperty method</a> (inherited from <code>QAMessage</code> )     | Gets an int message property                                                                                                                                            |
| <a href="#">GetLongProperty method</a> (inherited from <code>QAMessage</code> )    | Gets a long message property                                                                                                                                            |
| <a href="#">GetProperty method</a> (inherited from <code>QAMessage</code> )        | Gets a message property                                                                                                                                                 |
| <a href="#">GetPropertyNames method</a> (inherited from <code>QAMessage</code> )   | Gets an enumerator over the property names of the message                                                                                                               |
| <a href="#">GetPropertyType method</a> (inherited from <code>QAMessage</code> )    | Returns the property type of the given property                                                                                                                         |
| <a href="#">GetSbyteProperty method</a> (inherited from <code>QAMessage</code> )   | Gets a signed byte message property                                                                                                                                     |
| <a href="#">GetShortProperty method</a> (inherited from <code>QAMessage</code> )   | Gets a short message property                                                                                                                                           |
| <a href="#">GetStringProperty method</a> (inherited from <code>QAMessage</code> )  | Gets a string message property                                                                                                                                          |
| <a href="#">PropertyExists method</a> (inherited from <code>QAMessage</code> )     | Indicates whether the given property has been set for this message                                                                                                      |
| <a href="#">ReadText method</a>                                                    | Read unread text into the given buffer. Any additional unread text must be read by subsequent calls to this method. Text is read from the beginning of any unread text. |
| <a href="#">SetBooleanProperty method</a> (inherited from <code>QAMessage</code> ) | Sets a boolean property                                                                                                                                                 |
| <a href="#">SetDoubleProperty method</a> (inherited from <code>QAMessage</code> )  | Sets a double property                                                                                                                                                  |

| Member                                                              | Description                                                                                  |
|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <a href="#">SetFloatProperty method</a> (inherited from QAMessage)  | Sets a float property                                                                        |
| <a href="#">SetIntProperty method</a> (inherited from QAMessage)    | Sets an int property                                                                         |
| <a href="#">SetLongProperty method</a> (inherited from QAMessage)   | Sets a long property                                                                         |
| <a href="#">SetProperty method</a> (inherited from QAMessage)       | Sets a property. The property type must be one of the acceptable primitive types, or String. |
| <a href="#">SetSbyteProperty method</a> (inherited from QAMessage)  | Sets a byte property                                                                         |
| <a href="#">SetShortProperty method</a> (inherited from QAMessage)  | Sets a short property                                                                        |
| <a href="#">SetStringProperty method</a> (inherited from QAMessage) | Sets a string property                                                                       |
| <a href="#">WriteText method</a>                                    | Append text to the text of the message.                                                      |

Protected instance methods

| Member                                                    | Description                        |
|-----------------------------------------------------------|------------------------------------|
| <a href="#">Dispose method</a> (inherited from QAMessage) | Clean up any resources being used. |

## Text property

The message text. If the message exceeds the maximum message chunk size, then the Text value will be null. In the latter case, you should use the readText method to receive the text.

Prototypes

```
' Visual Basic
Public Property Text As String
```

```
// C#
public string Text {get;set;}
```

## TextLength property

The length, in characters, of the message

Prototypes

```
' Visual Basic
Public Readonly Property TextLength As Long

// C#
public long TextLength {get;}
```

## ReadText method

Read unread text into the given buffer. Any additional unread text must be read by subsequent calls to this method. Text is read from the beginning of any unread text.

Prototypes

```
' Visual Basic
Public Function ReadText( _
    ByVal buf As System.Text.StringBuilder _
) As Integer

// C#
public int ReadText(
    System.Text.string Builder buf
);
```

Parameters

- ◆ **buf** Target buffer for any read text

Return value

the number of characters read or -1 if there are no more characters to read

## WriteText method

Append text to the text of the message.

Prototypes

```
' Visual Basic
Public Sub WriteText( _
    ByVal val As String _
)

// C#
public void WriteText(
    string val
);
```

Parameters

- ◆ **val** the text to append

---

# QATransactionalManager class

The transactional QA manager. All message puts and gets done via this manager are done transactionally. That is, all puts and gets occur within a transaction and are all committed or rolled back together. There is always a transaction. Committing or rolling back a transaction implicitly begins a new transaction.

Prototypes

```
' Visual Basic
Public Class QATransactionalManager
    Inherits QAManagerBase

// C#
public class QATransactionalManager :
    QAManagerBase
```

## QATransactionalManager members

Public instance properties

| Member                                                                   | Description                                                                                                                          |
|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">LastError property</a> (inherited from QAManagerBase)        | The error code of the last executed method, with 0 indicating success.                                                               |
| <a href="#">LastErrorMessage property</a> (inherited from QAManagerBase) | The text of the error associated with the last executed method. Will be null if the last executed method result in a LastError of 0. |
| <a href="#">Mode property</a> (inherited from QAManagerBase)             | The acknowledgement mode for receiving all messages through this manager                                                             |

Public instance methods

| Member                                                               | Description                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">BrowseMessages method</a> (inherited from QAManagerBase) | Browse the next available messages waiting that have been sent to the given address. The messages are just being browsed, so they cannot be acknowledged. Enumerators returned from the same manager cannot have their method calls interlaced. Interlacing calls may result in messages meant for one iterator to be browsed in another iterator. |
| <a href="#">Close method</a> (inherited from QAManagerBase)          | Closes the connection to the QA message system and releases any resources. Any calls to close beyond the first are ignored. Any subsequent calls to a method, other than close, will result in an exception being thrown.                                                                                                                          |



| Member                                                                           | Description                                                                                                                                                                                                                                                                   |
|----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Commit method</a>                                                    | Commits the putting and getting of all uncommitted puts and gets of messages. A message put using a transactional manager will not be received until commit occurs. Similarly, the get of a message via a transactional manager will not appear gotten until a commit occurs. |
| <a href="#">CreateBinaryMessage method</a><br>(inherited from QAManagerBase)     | Create a BinaryMessage instance appropriate for sending.                                                                                                                                                                                                                      |
| <a href="#">CreateTextMessage method</a> (inherited from QAManagerBase)          | Create a TextMessage instance appropriate for sending.                                                                                                                                                                                                                        |
| <a href="#">GetBooleanStoreProperty method</a><br>(inherited from QAManagerBase) | Gets a boolean message store property                                                                                                                                                                                                                                         |
| <a href="#">GetDoubleStoreProperty method</a><br>(inherited from QAManagerBase)  | Gets a double message store property                                                                                                                                                                                                                                          |
| <a href="#">GetFloatStoreProperty method</a><br>(inherited from QAManagerBase)   | Gets a float message store property                                                                                                                                                                                                                                           |
| <a href="#">GetIntStoreProperty method</a> (inherited from QAManagerBase)        | Gets an int message store property                                                                                                                                                                                                                                            |
| <a href="#">GetLongStoreProperty method</a><br>(inherited from QAManagerBase)    | Gets a long message store property                                                                                                                                                                                                                                            |
| <a href="#">GetMessage method</a> (inherited from QAManagerBase)                 | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call blocks indefinitely until a message is available.                                                                                            |
| <a href="#">GetMessageNoWait method</a> (inherited from QAManagerBase)           | Get the next available message waiting that has been sent to the given address. If there is no message available, then it returns null immediately, without blocking.                                                                                                         |
| <a href="#">GetMessageTimeout method</a> (inherited from QAManagerBase)          | Get the next available message waiting that has been sent to the given address. If there is no message available, then this call will wait up to the timeout time until a message is available.                                                                               |
| <a href="#">GetSbyteStoreProperty method</a><br>(inherited from QAManagerBase)   | Gets a signed byte message store property                                                                                                                                                                                                                                     |
| <a href="#">GetShortStoreProperty method</a><br>(inherited from QAManagerBase)   | Gets a short message store property                                                                                                                                                                                                                                           |
| <a href="#">GetStoreProperty method</a> (inherited from QAManagerBase)           | Gets a message store property                                                                                                                                                                                                                                                 |
| <a href="#">GetStringStoreProperty method</a><br>(inherited from QAManagerBase)  | Gets a string message store property                                                                                                                                                                                                                                          |

| Member                                                                        | Description                                                                                                                                                                                                                                         |
|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Open method</a>                                                   | Open the transactional manager. The open method must be the first method called after creating a manager.                                                                                                                                           |
| <a href="#">PutMessage method</a> (inherited from QAManagerBase)              | Puts the message into the message system addressed to the given address.                                                                                                                                                                            |
| <a href="#">PutMessageTimeToLive method</a> (inherited from QAManagerBase)    | Puts the message into the message system addressed to the given address, with the given time-to-live.                                                                                                                                               |
| <a href="#">Rollback method</a>                                               | Rolls back the putting and getting of all uncommitted puts and gets of messages. A message put using a transactional manager will not be sent. Similarly, the get of a message via a transactional manager will not appear to have not been gotten. |
| <a href="#">SetBooleanStoreProperty method</a> (inherited from QAManagerBase) | Sets a boolean message store property                                                                                                                                                                                                               |
| <a href="#">SetDoubleStoreProperty method</a> (inherited from QAManagerBase)  | Sets a double message store property                                                                                                                                                                                                                |
| <a href="#">SetFloatStoreProperty method</a> (inherited from QAManagerBase)   | Sets a float message store property                                                                                                                                                                                                                 |
| <a href="#">SetIntStoreProperty method</a> (inherited from QAManagerBase)     | Sets an int message store property                                                                                                                                                                                                                  |
| <a href="#">SetLongStoreProperty method</a> (inherited from QAManagerBase)    | Sets a long message store property                                                                                                                                                                                                                  |
| <a href="#">SetMessageListener method</a> (inherited from QAManagerBase)      | Sets a listener for messages available for the given address. Only one listener can be set for a given address. Setting with a null listener clears out any listener for that address.                                                              |
| <a href="#">SetProperty method</a> (inherited from QAManagerBase)             | Sets the named property to the given value. Properties for this QAManagerBase may be set with this method as an alternative to the properties file at creation. Properties must be set before calling the open() methods of the derived classes.    |
| <a href="#">SetSbyteStoreProperty method</a> (inherited from QAManagerBase)   | Sets a byte message store property                                                                                                                                                                                                                  |
| <a href="#">SetShortStoreProperty method</a> (inherited from QAManagerBase)   | Sets a short message store property                                                                                                                                                                                                                 |
| <a href="#">SetStoreProperty method</a> (inherited from QAManagerBase)        | Sets a message store property. The property type must be one of the acceptable primitive types, or String.                                                                                                                                          |
| <a href="#">SetStringStoreProperty method</a> (inherited from QAManagerBase)  | Sets a string message store property                                                                                                                                                                                                                |

| Member                                                                   | Description                                                                                                                                                                                                                        |
|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Start method</a> (inherited from QAManagerBase)              | Once started the manager will receive any incoming messages. Any calls to start beyond the first without an intervening stop are ignored.                                                                                          |
| <a href="#">Stop method</a> (inherited from QA-ManagerBase)              | Once stopped the manager will not receive any incoming messages. The messages are not lost. They just won't be received until the manager is started. Any calls to stop beyond the first without an intervening start are ignored. |
| <a href="#">TriggerSendReceive method</a> (inherited from QAManagerBase) | Causes a synchronization with the QA message server, uploading any messages not meant for this client, and downloading any messages meant for this client.                                                                         |

Protected static fields  
(Shared)

| Member                                                       | Description                                    |
|--------------------------------------------------------------|------------------------------------------------|
| <a href="#">isOpen field</a> (inherited from QA-ManagerBase) | Indicates whether instance is in an open state |
| <a href="#">mgrBase field</a> (inherited from QAManagerBase) | Handle to the underlying c++ qa manager        |

Protected instance fields

| Member                                                       | Description                                    |
|--------------------------------------------------------------|------------------------------------------------|
| <a href="#">isOpen field</a> (inherited from QA-ManagerBase) | Indicates whether instance is in an open state |
| <a href="#">mgrBase field</a> (inherited from QAManagerBase) | Handle to the underlying c++ qa manager        |

Protected instance methods

| Member                         | Description                        |
|--------------------------------|------------------------------------|
| <a href="#">Dispose method</a> | Clean up any resources being used. |

## Commit method

Commits the putting and getting of all uncommitted puts and gets of

---

messages. A message put using a transactional manager will not be received until commit occurs. Similarly, the get of a message via a transactional manager will not appear gotten until a commit occurs.

Prototypes

```
' Visual Basic  
Public Sub Commit()
```

```
// C#  
public void Commit();
```

Exceptions

◆ [QAEException class](#) - if there is a problem committing

## Dispose method

Clean up any resources being used.

Prototypes

```
' Visual Basic  
Overloads Overrides Protected Sub Dispose( _  
    ByVal disposing As Boolean _  
    )
```

```
// C#  
protected override void Dispose(  
    bool disposing  
    );
```

Parameters

◆ **disposing** true to release both managed and unmanaged resources; false to release only unmanaged resources.

## Open method

Open the transactional manager. The open method must be the first method called after creating a manager.

Prototypes

```
' Visual Basic  
Public Sub Open()
```

```
// C#  
public void Open();
```

Exceptions

◆ [QAEException class](#) - if there is a problem opening the manager

## Rollback method

Rolls back the putting and getting of all uncommitted puts and gets of messages. A message put using a transactional manager will not be sent. Similarly, the get of a message via a transactional manager will not appear to have not been gotten.

Prototypes

```
' Visual Basic  
Public Sub Rollback()
```

```
// C#  
public void Rollback();
```

Exceptions

- ◆ [QAException class](#) - if there is a problem rolling back



---

# Index

## Symbols

- ~QABinaryMessage function
  - QAnywhere C++ API 134
- ~QAError function
  - QAnywhere C++ API 137
- ~QAManager function
  - QAnywhere C++ API 140
- ~QAManagerBase function
  - QAnywhere C++ API 152
- ~QAManagerFactory function
  - QAnywhere C++ API 154
- ~QAMessage function
  - QAnywhere C++ API 166
- ~QAMessageListener function
  - QAnywhere C++ API 167
- ~QATextMessage function
  - QAnywhere C++ API 170
- ~QATransactionalManager function
  - QAnywhere C++ API 172
- A**
  - ABS\_RETRY\_TIMEOUT field
    - iAnywhere.QAnywhere.Client namespace 176
  - ABS\_RETRY\_TIMEOUT variable
    - QAnywhere C++ API 123
  - acknowledge function
    - QAnywhere C++ API 139
  - Acknowledge method
    - iAnywhere.QAnywhere.Client namespace 199
  - acknowledgeAll function
    - QAnywhere C++ API 139
  - AcknowledgeAll method
    - iAnywhere.QAnywhere.Client namespace 199
  - acknowledgement modes
    - QAManager class (C++) 61
    - QAManager class (.NET) 63
  - AcknowledgementMode class
    - QAnywhere C++ API 122
  - AcknowledgementMode enumeration
    - iAnywhere.QAnywhere.Client namespace 174
  - acknowledgeUntil function
    - QAnywhere C++ API 139
  - AcknowledgeUntil method
    - iAnywhere.QAnywhere.Client namespace 199
  - ADAPTER field
    - iAnywhere.QAnywhere.Client namespace 176
  - ADAPTER variable
    - QAnywhere C++ API 123
  - adding client messages store IDs
    - QAnywhere 34
  - Address property
    - iAnywhere.QAnywhere.Client namespace 225
  - addresses
    - QAnywhere 59
    - QAnywhere JMS Connector 59
    - setting in QAnywhere messages (C++) 67
    - setting in QAnywhere messages (.NET) 68
  - addressing JMS messages meant for QAnywhere
    - about 48
  - addressing QAnywhere messages meant for JMS
    - about 46
  - application-to-application messaging *see also* messaging
    - QAnywhere 2
  - architecture
    - integrating messaging into a data synchronization application 51
    - QAnywhere 5
  - asynchronous message receipt
    - QAnywhere 70
  - authentication
    - QAnywhere 99
  - automatic policy
    - QAnywhere Agent 89
- B**
  - BodyLength property
    - iAnywhere.QAnywhere.Client namespace 184
  - BrowseMessages method
    - iAnywhere.QAnywhere.Client namespace 205
- C**
  - c option

|                                              |     |                                               |     |
|----------------------------------------------|-----|-----------------------------------------------|-----|
| QAnywhere Agent [qaagent]                    | 82  | COMMON_MSG_STORE_ERROR variable               |     |
| castToBinaryMessage function                 |     | QAnywhere C++ API                             | 136 |
| QAnywhere C++ API                            | 156 | COMMON_MSG_STORE_NOT_INITIALIZED variable     |     |
| castToTextMessage function                   |     | QAnywhere C++ API                             | 136 |
| QAnywhere C++ API                            | 156 | COMMON_MSG_STORE_TOO_LARGE variable           |     |
| clearProperties function                     |     | QAnywhere C++ API                             | 136 |
| QAnywhere C++ API                            | 156 | COMMON_NO_DEST_ERROR variable                 |     |
| ClearProperties method                       |     | QAnywhere C++ API                             | 136 |
| iAnywhere.QAnywhere.Client namespace         | 227 | COMMON_NO_IMPLEMENTATION variable             |     |
| client message store IDs                     |     | QAnywhere C++ API                             | 137 |
| adding QAnywhere to the server message store | 34  | COMMON_OPEN_ERROR variable                    |     |
| client message stores                        |     | QAnywhere C++ API                             | 137 |
| client store properties                      | 113 | COMMON_OPEN_LOG_FILE_ERROR variable           |     |
| creating the IDs                             | 35  | QAnywhere C++ API                             | 137 |
| custom client store properties               | 114 | COMMON_TERMINATE_ERROR variable               |     |
| encrypting QAnywhere                         | 97  | QAnywhere C++ API                             | 137 |
| initializing with -si option                 | 91  | COMMON_UNEXPECTED_EOM_ERROR variable          |     |
| pre-defined client store properties          | 113 | QAnywhere C++ API                             | 137 |
| QAnywhere architecture                       | 6   | communication streams                         |     |
| QAnywhere security                           | 96  | encrypting QAnywhere                          | 98  |
| QAnywhere tutorial                           | 16  | COMPRESSED field                              |     |
| setting up in QAnywhere                      | 35  | iAnywhere.QAnywhere.Client namespace          | 177 |
| client store properties                      |     | compression                                   |     |
| QAnywhere                                    | 113 | QAnywhere JMS connectors                      | 44  |
| QAnywhere custom                             | 114 | COMPRESSION_LEVEL property                    |     |
| QAnywhere pre-defined                        | 113 | QAManager properties                          | 66  |
| close function                               |     | condition syntax                              |     |
| QAnywhere C++ API                            | 142 | QAnywhere                                     | 106 |
| Close method                                 |     | conditions                                    |     |
| iAnywhere.QAnywhere.Client namespace         | 205 | QAnywhere schedule syntax                     | 106 |
| commit function                              |     | configuring multiple connectors               |     |
| QAnywhere C++ API                            | 172 | QAnywhere                                     | 45  |
| Commit method                                |     | configuring the JMS connector properties file |     |
| iAnywhere.QAnywhere.Client namespace         | 245 | QAnywhere                                     | 43  |
| COMMON_GET_INIT_FILE_ERROR variable          |     | connectors                                    |     |
| QAnywhere C++ API                            | 135 | configuring multiple QAnywhere                | 45  |
| COMMON_INIT_ERROR variable                   |     | QAnywhere JMS connector properties files      | 43  |
| QAnywhere C++ API                            | 135 | conventions                                   |     |
| COMMON_INIT_THREAD_ERROR variable            |     | documentation                                 | x   |
| QAnywhere C++ API                            | 136 | createBinaryMessage function                  |     |
| COMMON_INVALID_PROPERTY variable             |     | QAnywhere C++ API                             | 142 |
| QAnywhere C++ API                            | 136 | CreateBinaryMessage method                    |     |
| COMMON_MSG_NOT_WRITEABLE_ERROR variable      |     | iAnywhere.QAnywhere.Client namespace          | 206 |
| QAnywhere C++ API                            | 136 | createQAManager function                      |     |
| COMMON_MSG_RETRIEVE_ERROR variable           |     | QAnywhere C++ API                             | 153 |
| QAnywhere C++ API                            | 136 |                                               |     |



|                                        |      |                                      |      |
|----------------------------------------|------|--------------------------------------|------|
| CreateQAManager method                 |      | conventions                          | x    |
| iAnywhere.QAnywhere.Client namespace   | 222  | SQL Anywhere Studio                  | viii |
| createQATransactionalManager function  |      | <b>E</b>                             |      |
| QAnywhere C++ API                      | 153  | EAServer                             |      |
| CreateQATransactionalManager method    |      | QAnywhere and                        | 8    |
| iAnywhere.QAnywhere.Client namespace   | 222  | encrypting                           |      |
| createTextMessage function             |      | QAnywhere client message stores      | 97   |
| QAnywhere C++ API                      | 142  | QAnywhere communication stream       | 98   |
| CreateTextMessage method               |      | encrypting the client message store  |      |
| iAnywhere.QAnywhere.Client namespace   | 206  | QAnywhere                            | 97   |
| QAManager class                        | 67   | encrypting the communication stream  |      |
| createTextMessage method               |      | QAnywhere                            | 98   |
| QAManager class                        | 67   | ErrorCode property                   |      |
| creating                               |      | iAnywhere.QAnywhere.Client namespace | 194  |
| QAnywhere server message store         | 32   | Expiration property                  |      |
| creating a secure client message store |      | iAnywhere.QAnywhere.Client namespace | 226  |
| QAnywhere                              | 96   | EXPLICIT_ACKNOWLEDGEMENT variable    |      |
| creating client message store IDs      |      | QAnywhere C++ API                    | 122  |
| QAnywhere                              | 35   | <b>F</b>                             |      |
| custom client store properties         |      | failover                             |      |
| QAnywhere                              | 114  | QAnywhere                            | 53   |
| <b>D</b>                               |      | feedback                             |      |
| dbeng9                                 |      | documentation                        | xiii |
| QAnywhere Agent and                    | 80   | providing                            | xiii |
| dblsn                                  |      | Finalize method                      |      |
| QAnywhere Agent and                    | 80   | iAnywhere.QAnywhere.Client namespace | 223  |
| dbmsync utility                        |      | FROM_ADDR field                      |      |
| QAnywhere Agent and                    | 80   | iAnywhere.QAnywhere.Client namespace | 177  |
| DEFAULT_PRIORITY variable              |      | FROM_ADDR variable                   |      |
| QAnywhere C++ API                      | 156  | QAnywhere C++ API                    | 123  |
| DEFAULT_TIME_TO_LIVE variable          |      | <b>G</b>                             |      |
| QAnywhere C++ API                      | 156  | getAddress function                  |      |
| delete rules                           |      | QAnywhere C++ API                    | 157  |
| QAnywhere                              | 118  | getBodyLength function               |      |
| deleteMessage function                 |      | QAnywhere C++ API                    | 129  |
| QAnywhere C++ API                      | 142  | getBooleanProperty function          |      |
| deleteQAManager function               |      | QAnywhere C++ API                    | 157  |
| QAnywhere C++ API                      | 153  | GetBooleanProperty method            |      |
| deleteQATransactionalManager function  |      | iAnywhere.QAnywhere.Client namespace | 228  |
| QAnywhere C++ API                      | 154  | getBooleanStoreProperty function     |      |
| deploying QAnywhere                    |      | QAnywhere C++ API                    | 143  |
| about                                  | 78   | GetBooleanStoreProperty method       |      |
| Dispose method                         |      | iAnywhere.QAnywhere.Client namespace | 206  |
| iAnywhere.QAnywhere.Client namespace   | 200, |                                      |      |
| 206, 223, 227, 246                     |      |                                      |      |
| documentation                          |      |                                      |      |

|                                      |          |                                      |     |
|--------------------------------------|----------|--------------------------------------|-----|
| getBytesProperty function            |          | iAnywhere.QAnywhere.Client namespace | 209 |
| QAnywhere C++ API                    | 157      | getMessageID function                |     |
| getBytesStoreProperty function       |          | QAnywhere C++ API                    | 159 |
| QAnywhere C++ API                    | 143      | getMessageNoWait function            |     |
| getDoubleProperty function           |          | QAnywhere C++ API                    | 145 |
| QAnywhere C++ API                    | 157      | getMessageNoWait method              |     |
| GetDoubleProperty method             |          | iAnywhere.QAnywhere.Client namespace | 209 |
| iAnywhere.QAnywhere.Client namespace | 228      | getMessageTimeout function           |     |
| getDoubleStoreProperty function      |          | QAnywhere C++ API                    | 145 |
| QAnywhere C++ API                    | 143      | getMessageTimeout method             |     |
| GetDoubleStoreProperty method        |          | iAnywhere.QAnywhere.Client namespace | 209 |
| iAnywhere.QAnywhere.Client namespace | 207      | getMode function                     |     |
| getExpiration function               |          | QAnywhere C++ API                    | 146 |
| QAnywhere C++ API                    | 158      | getPriority function                 |     |
| getFloatProperty function            |          | QAnywhere C++ API                    | 159 |
| QAnywhere C++ API                    | 158      | GetProperty method                   |     |
| GetFloatProperty method              |          | iAnywhere.QAnywhere.Client namespace | 230 |
| iAnywhere.QAnywhere.Client namespace | 229      | getPropertyNames function            |     |
| getFloatStoreProperty function       |          | QAnywhere C++ API                    | 159 |
| QAnywhere C++ API                    | 144      | GetPropertyNames method              |     |
| GetFloatStoreProperty method         |          | iAnywhere.QAnywhere.Client namespace | 230 |
| iAnywhere.QAnywhere.Client namespace | 207      | getPropertyType function             |     |
| getInReplyToID function              |          | QAnywhere C++ API                    | 160 |
| QAnywhere C++ API                    | 158      | GetPropertyType method               |     |
| getIntProperty function              |          | iAnywhere.QAnywhere.Client namespace | 230 |
| QAnywhere C++ API                    | 158      | getRedelivered function              |     |
| GetIntProperty method                |          | QAnywhere C++ API                    | 160 |
| iAnywhere.QAnywhere.Client namespace | 229      | getReplyToAddress function           |     |
| getIntStoreProperty function         |          | QAnywhere C++ API                    | 160 |
| QAnywhere C++ API                    | 144      | GetSbyteProperty method              |     |
| GetIntStoreProperty method           |          | iAnywhere.QAnywhere.Client namespace | 231 |
| iAnywhere.QAnywhere.Client namespace | 208      | GetSbyteStoreProperty method         |     |
| getLastError function                |          | iAnywhere.QAnywhere.Client namespace | 210 |
| QAnywhere C++ API                    | 144, 154 | getShortProperty function            |     |
| getLastErrorMsg function             |          | QAnywhere C++ API                    | 160 |
| QAnywhere C++ API                    | 144, 154 | getShortProperty method              |     |
| getLongProperty function             |          | iAnywhere.QAnywhere.Client namespace | 231 |
| QAnywhere C++ API                    | 159      | getShortStoreProperty function       |     |
| GetLongProperty method               |          | QAnywhere C++ API                    | 146 |
| iAnywhere.QAnywhere.Client namespace | 229      | getShortStoreProperty method         |     |
| getLongStoreProperty function        |          | iAnywhere.QAnywhere.Client namespace | 210 |
| QAnywhere C++ API                    | 145      | getStoreProperty method              |     |
| GetLongStoreProperty method          |          | iAnywhere.QAnywhere.Client namespace | 211 |
| iAnywhere.QAnywhere.Client namespace | 208      | getStringProperty function           |     |
| getMessage function                  |          | QAnywhere C++ API                    | 161 |
| QAnywhere C++ API                    | 145      | GetStringProperty method             |     |
| GetMessage method                    |          | iAnywhere.QAnywhere.Client namespace | 232 |

- 
- getStringStoreProperty function
    - QAnywhere C++ API 146
  - GetStringStoreProperty method
    - iAnywhere.QAnywhere.Client namespace 211
  - getText function
    - QAnywhere C++ API 169
  - getTextLength function
    - QAnywhere C++ API 169
  - getTimestamp function
    - QAnywhere C++ API 161
  - getTimestampAsString function
    - QAnywhere C++ API 162
  - getting started
    - QAnywhere 10
  - H**
  - handling push notifications and network status changes
    - QAnywhere 73
  - I**
  - i anywhere.connector.address property
    - QAnywhere 44
  - i anywhere.connector.compressionLevel property
    - QAnywhere 44
  - i anywhere.connector.id property
    - QAnywhere 44
  - i anywhere.connector.logLevel property
    - QAnywhere 44
  - i anywhere.connector.NativeConnection property
    - QAnywhere 44
  - i anywhere.connector.outgoing.deadMessageAddress property
    - QAnywhere 44
  - iAnywhere.QAnywhere.Client namespace
    - iAnywhere.QAnywhere.Client namespace 173
  - icons
    - used in manuals xii
  - id option
    - QAnywhere Agent [qaagent] 84
  - IDs
    - adding QAnywhere to the server message store 34
    - understanding QAnywhere addresses 59
  - implementing transactional messaging
    - about 75
  - IMPLICIT\_ACKNOWLEDGEMENT variable
    - QAnywhere C++ API 122
  - initializing the QAnywhere client API
    - about 60
  - InReplyToID property
    - iAnywhere.QAnywhere.Client namespace 226
  - Instance property
    - iAnywhere.QAnywhere.Client namespace 221
  - InstanceCount property
    - iAnywhere.QAnywhere.Client namespace 221
  - InstanceID field
    - iAnywhere.QAnywhere.Client namespace 221
  - introduction to QAnywhere 1
  - isOpen field
    - iAnywhere.QAnywhere.Client namespace 204
  - iu option
    - QAnywhere Agent [qaagent] 85
  - J**
  - JMS
    - running MobiLink with messaging and a JMS Connector 42
  - JMS Connector
    - properties file 43
    - QAnywhere addresses 59
    - QAnywhere architecture 9
  - JMS properties
    - mapping JMS messages on to QAnywhere messages 50
  - JMS providers
    - QAnywhere architecture 8
  - L**
  - la option
    - QAnywhere Agent [qaagent] 85
  - LastError property
    - iAnywhere.QAnywhere.Client namespace 204, 222
  - LastErrorMessage property
    - iAnywhere.QAnywhere.Client namespace 204, 222
  - Listener utility
    - QAnywhere Agent and 8, 80
  - LOG\_FILE property
    - QAManager properties 66
  - M**
  - manage client message store passwords

|                                               |                           |                                                   |      |
|-----------------------------------------------|---------------------------|---------------------------------------------------|------|
| QAnywhere                                     | 96                        | messaging systems                                 |      |
| mapping JMS messages on to QAnywhere messages |                           | JMS integration with QAnywhere                    | 42   |
| about                                         | 49                        | messaging with external messaging systems         |      |
| mapping QAnywhere messages on to JMS messages |                           | QAnywhere architecture                            | 8    |
| about                                         | 47                        | messaging with push notifications                 |      |
| MAX_IN_MEMORY_MESSAGE_SIZE property           |                           | QAnywhere architecture                            | 6    |
| QAManager properties                          | 66                        | mgrBase field                                     |      |
| message addresses                             |                           | iAnywhere.QAnywhere.Client namespace              | 204  |
| QAnywhere                                     | 59                        | MobiLink                                          |      |
| message headers                               |                           | integrating messaging into a data synchronization |      |
| QAnywhere transmission rules                  | 110                       | application                                       | 51   |
| message listeners                             |                           | MobiLink synchronization server                   |      |
| QAnywhere                                     | 70                        | QAnywhere                                         | 33   |
| message properties                            |                           | MobiLink with messaging                           |      |
| QAnywhere                                     | 111                       | QAnywhere setup                                   | 31   |
| message properties file                       |                           | QAnywhere tutorial                                | 13   |
| QAnywhere                                     | 43                        | simple messaging architecture                     | 6    |
| message store IDs                             |                           | starting                                          | 33   |
| adding QAnywhere to the server message store  | 34                        | Mode property                                     |      |
| message stores                                |                           | iAnywhere.QAnywhere.Client namespace              | 205  |
| encrypting QAnywhere client message stores    | 97                        | -mp option                                        |      |
| QAnywhere client architecture                 | 6                         | QAnywhere Agent [qaagent]                         | 85   |
| QAnywhere server architecture                 | 5                         | MSG_TYPE field                                    |      |
| message transmission                          |                           | iAnywhere.QAnywhere.Client namespace              | 177  |
| QAnywhere                                     | 101                       | MSG_TYPE variable                                 |      |
| MessageID property                            |                           | QAnywhere C++ API                                 | 123  |
| iAnywhere.QAnywhere.Client namespace          | 226                       | <b>N</b>                                          |      |
| MessageListener class                         |                           | NETWORK field                                     |      |
| QAnywhere (C++)                               | 71                        | iAnywhere.QAnywhere.Client namespace              | 177  |
| QAnywhere system messages                     | 73                        | network status                                    |      |
| MessageProperties class                       |                           | handling changes in QAnywhere                     | 73   |
| iAnywhere.QAnywhere.Client namespace          | 175                       | NETWORK variable                                  |      |
| QAnywhere C++ API                             | 123                       | QAnywhere C++ API                                 | 124  |
| MessageProperties constructor                 |                           | NETWORK_STATUS field                              |      |
| iAnywhere.QAnywhere.Client namespace          | 176                       | iAnywhere.QAnywhere.Client namespace              | 177  |
| messages stores                               |                           | NETWORK_STATUS variable                           |      |
| creating QAnywhere client message stores      | 35                        | QAnywhere C++ API                                 | 124  |
| creating the QAnywhere server message store   | 32                        | NETWORK_STATUS_NOTIFICATION variable              |      |
| MessageType class                             |                           | QAnywhere C++ API                                 | 126  |
| QAnywhere C++ API                             | 126                       | newsgroups                                        |      |
| MessageType enumeration                       |                           | technical support                                 | xiii |
| iAnywhere.QAnywhere.Client namespace          | 180                       | notifications                                     |      |
| messaging                                     | <i>see also</i> QAnywhere | disabling QAnywhere                               | 90   |
| application-to-application                    | 2                         | handling in QAnywhere                             | 73   |
| QAnywhere addresses                           | 59                        | QAnywhere introduction to                         | 40   |
| QAnywhere features                            | 3                         | QAnywhere with SMS                                | 40   |
| QAnywhere quick start                         | 10                        |                                                   |      |

**O**

|                                      |          |
|--------------------------------------|----------|
| -o option                            |          |
| QAnywhere Agent [qaagent]            | 86       |
| -on option                           |          |
| QAnywhere Agent [qaagent]            | 86       |
| ondemand policy                      |          |
| QAnywhere Agent                      | 88       |
| onMessage function                   |          |
| QAnywhere C++ API                    | 167      |
| onMessage method                     |          |
| QAManager class (C++)                | 70       |
| open function                        |          |
| QAnywhere C++ API                    | 139, 172 |
| Open method                          |          |
| iAnywhere.QAnywhere.Client namespace | 200, 246 |
| -os option                           |          |
| QAnywhere Agent [qaagent]            | 87       |
| -ot option                           |          |
| QAnywhere Agent [qaagent]            | 87       |

**P**

|                                       |     |
|---------------------------------------|-----|
| password authentication with MobiLink |     |
| QAnywhere applications                | 99  |
| peekFirstMessage function             |     |
| QAnywhere C++ API                     | 147 |
| peekNextMessage function              |     |
| QAnywhere C++ API                     | 147 |
| persistence                           |     |
| QAnywhere messages                    | 118 |
| policies                              |     |
| QAnywhere                             | 37  |
| QAnywhere architecture                | 6   |
| QAnywhere tutorial                    | 19  |
| -policy option                        |     |
| QAnywhere Agent [qaagent]             | 88  |
| -port option                          |     |
| QAnywhere Agent [qaagent]             | 90  |
| pre-defined client store properties   |     |
| QAnywhere                             | 113 |
| Priority property                     |     |
| iAnywhere.QAnywhere.Client namespace  | 226 |
| properties                            |     |
| QAManager                             | 66  |
| propertyExists function               |     |
| QAnywhere C++ API                     | 162 |

|                                      |     |
|--------------------------------------|-----|
| PropertyExists method                |     |
| iAnywhere.QAnywhere.Client namespace | 232 |
| publishMessage function              |     |
| QAnywhere C++ API                    | 147 |
| push notifications                   |     |
| QAnywhere introduction to            | 40  |
| -push option                         |     |
| QAnywhere Agent [qaagent]            | 90  |
| PUSH_NOTIFICATION variable           |     |
| QAnywhere C++ API                    | 126 |
| putMessage function                  |     |
| QAnywhere C++ API                    | 147 |
| PutMessage method                    |     |
| iAnywhere.QAnywhere.Client namespace | 212 |
| putMessageTimeToLive function        |     |
| QAnywhere C++ API                    | 148 |
| PutMessageTimeToLive method          |     |
| iAnywhere.QAnywhere.Client namespace | 212 |

**Q**

|                                      |     |
|--------------------------------------|-----|
| -q option                            |     |
| QAnywhere Agent [qaagent]            | 90  |
| QA_NO_ERROR variable                 |     |
| QAnywhere C++ API                    | 137 |
| qaagent utility                      |     |
| about                                | 37  |
| syntax                               | 80  |
| QABinaryMessage class                |     |
| iAnywhere.QAnywhere.Client namespace | 181 |
| instantiating (C++)                  | 67  |
| instantiating (.NET)                 | 67  |
| QAnywhere C++ API                    | 127 |
| QAEError class                       |     |
| QAnywhere C++ API                    | 135 |
| QAException class                    |     |
| iAnywhere.QAnywhere.Client namespace | 192 |
| QAException constructor              |     |
| iAnywhere.QAnywhere.Client namespace | 193 |
| qa.hpp                               |     |
| QAnywhere header file                | 60  |
| QAManager class                      |     |
| acknowledgement modes (C++)          | 61  |
| acknowledgement modes (.NET)         | 63  |
| iAnywhere.QAnywhere.Client namespace | 195 |
| initializing (C++)                   | 61  |
| initializing (.NET)                  | 63  |
| instantiating (C++)                  | 60  |

|                                                   |        |                                      |                                 |
|---------------------------------------------------|--------|--------------------------------------|---------------------------------|
| instantiating (.NET)                              | 62     | about                                | 37                              |
| QAnywhere C++ API                                 | 138    | simple messaging architecture        | 6                               |
| QAManager properties                              |        | syntax                               | 80                              |
| COMPRESSION_LEVEL                                 | 66     | QAnywhere architecture               |                                 |
| listed                                            | 66     | about                                | 5                               |
| LOG_FILE                                          | 66     | QAnywhere C++ API                    |                                 |
| MAX_IN_MEMORY_MESSAGE_SIZE                        | 66     | introduction                         | 56                              |
| properties file                                   | 60, 62 | QAnywhere client                     |                                 |
| RECEIVER_INTERVAL                                 | 66     | shutting down                        | 77                              |
| setting                                           | 64     | QAnywhere client API                 |                                 |
| QAManagerBase class                               |        | initializing                         | 60                              |
| iAnywhere.QAnywhere.Client namespace              | 201    | QAnywhere client applications        |                                 |
| QAnywhere C++ API                                 | 141    | writing                              | 55                              |
| QAManagerBase.MessageListener delegate            |        | QAnywhere clients                    |                                 |
| iAnywhere.QAnywhere.Client namespace              | 219    | introduction                         | 6                               |
| QAManagerFactory class                            |        | QAnywhere header file                |                                 |
| iAnywhere.QAnywhere.Client namespace              | 220    | qa.hpp                               | 60                              |
| initializing (C++)                                | 60     | QAnywhere Manager properties         | <i>see</i> QAManager            |
| initializing (.NET)                               | 62     | properties                           |                                 |
| QAnywhere C++ API                                 | 153    | QAnywhere namespace                  |                                 |
| QAMessage class                                   |        | including                            | 62                              |
| iAnywhere.QAnywhere.Client namespace              | 224    | QAnywhere .NET API                   |                                 |
| QAnywhere C++ API                                 | 155    | introduction                         | 56                              |
| QAMessageListener class                           |        | QAnywhere Notifier                   |                                 |
| QAnywhere C++ API                                 | 167    | architecture                         | 7                               |
| QAnywhere                                         |        | QAnywhere properties                 | <i>see</i> QAManager properties |
| about                                             | 1      | mapping QAnywhere messages on to JMS |                                 |
| addresses                                         | 59     | messages                             | 48                              |
| architecture                                      | 5      | QAnywhere transmission rules         |                                 |
| connecting to the client message store            | 82     | about                                | 101                             |
| delete rules                                      | 118    | QAPROPERTYTYPE enumeration           |                                 |
| deploying applications                            | 78     | iAnywhere.QAnywhere.Client namespace | 237                             |
| failover                                          | 53     | gastop utility                       |                                 |
| features                                          | 3      | use with qaagent -q (quiet mode)     | 90                              |
| integrating messaging into a data synchronization |        | use with qaagent -qi (quiet mode)    | 91                              |
| application                                       | 51     | QATextMessage class                  |                                 |
| programming interfaces                            | 56     | iAnywhere.QAnywhere.Client namespace | 238                             |
| quick start                                       | 10     | instantiating (C++)                  | 67                              |
| receiving notifications                           | 70     | instantiating (.NET)                 | 67                              |
| security                                          | 95     | QAnywhere C++ API                    | 168                             |
| setting up client-side components                 | 35     | QATransactionalManager class         |                                 |
| setting up server-side components                 | 32     | iAnywhere.QAnywhere.Client namespace | 242                             |
| transmission rules                                | 101    | QAnywhere (C++)                      | 75                              |
| transmission rules variables                      | 110    | QAnywhere C++ API                    | 171                             |
| tutorial                                          | 11     | -qi option                           |                                 |
| using JMS connectors                              | 42     | QAnywhere Agent [qaagent]            | 91                              |
| QAnywhere Agent                                   |        | queues                               |                                 |

|                                      |     |  |  |
|--------------------------------------|-----|--|--|
| understanding QAnywhere addresses    | 59  |  |  |
| quick start                          |     |  |  |
| QAnywhere                            | 10  |  |  |
| quiet mode                           |     |  |  |
| QAnywhere Agent [qaagent] -q         | 90  |  |  |
| QAnywhere Agent [qaagent] -qi        | 91  |  |  |
| <b>R</b>                             |     |  |  |
| readBinary function                  |     |  |  |
| QAnywhere C++ API                    | 129 |  |  |
| ReadBinary method                    |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 184 |  |  |
| readBoolean function                 |     |  |  |
| QAnywhere C++ API                    | 129 |  |  |
| ReadBoolean method                   |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 185 |  |  |
| readByte function                    |     |  |  |
| QAnywhere C++ API                    | 129 |  |  |
| readChar function                    |     |  |  |
| QAnywhere C++ API                    | 130 |  |  |
| ReadChar method                      |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 185 |  |  |
| readDouble function                  |     |  |  |
| QAnywhere C++ API                    | 130 |  |  |
| ReadDouble method                    |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 186 |  |  |
| readFloat function                   |     |  |  |
| QAnywhere C++ API                    | 130 |  |  |
| ReadFloat method                     |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 186 |  |  |
| reading                              |     |  |  |
| QAnywhere large messages             | 72  |  |  |
| reading very large messages          |     |  |  |
| about                                | 72  |  |  |
| readInt function                     |     |  |  |
| QAnywhere C++ API                    | 130 |  |  |
| ReadInt method                       |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 186 |  |  |
| readLong function                    |     |  |  |
| QAnywhere C++ API                    | 131 |  |  |
| ReadLong method                      |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 186 |  |  |
| ReadSbyte method                     |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 187 |  |  |
| readShort function                   |     |  |  |
| QAnywhere C++ API                    | 131 |  |  |
| ReadShort method                     |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 187 |  |  |
| readString function                  |     |  |  |
| QAnywhere C++ API                    | 131 |  |  |
| ReadString method                    |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 187 |  |  |
| readText function                    |     |  |  |
| QAnywhere C++ API                    | 169 |  |  |
| ReadText method                      |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 241 |  |  |
| receiving messages                   |     |  |  |
| QAnywhere (C++)                      | 69  |  |  |
| QAnywhere (.NET)                     | 69  |  |  |
| receiving messages asynchronously    |     |  |  |
| QAnywhere                            | 70  |  |  |
| receiving messages synchronously     |     |  |  |
| QAnywhere                            | 69  |  |  |
| recover function                     |     |  |  |
| QAnywhere C++ API                    | 140 |  |  |
| Recover method                       |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 200 |  |  |
| Redelivered property                 |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 227 |  |  |
| REGULAR variable                     |     |  |  |
| QAnywhere C++ API                    | 126 |  |  |
| ReplyToAddress property              |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 227 |  |  |
| reset function                       |     |  |  |
| QAnywhere C++ API                    | 131 |  |  |
| Reset method                         |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 188 |  |  |
| RETRY_FAILED field                   |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 178 |  |  |
| RETRY_FAILED variable                |     |  |  |
| QAnywhere C++ API                    | 124 |  |  |
| RETRY_FAILED_ADDR field              |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 178 |  |  |
| RETRY_FAILED_ADDR variable           |     |  |  |
| QAnywhere C++ API                    | 124 |  |  |
| RETRY_FAILED_PRIORITY field          |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 178 |  |  |
| RETRY_FAILED_PRIORITY variable       |     |  |  |
| QAnywhere C++ API                    | 124 |  |  |
| RETRY_MAX field                      |     |  |  |
| iAnywhere.QAnywhere.Client namespace | 178 |  |  |
| RETRY_MAX variable                   |     |  |  |
| QAnywhere C++ API                    | 124 |  |  |
| RETRY_TIMEOUT field                  |     |  |  |

|                                                        |                               |                                      |     |
|--------------------------------------------------------|-------------------------------|--------------------------------------|-----|
| iAnywhere.QAnywhere.Client namespace                   | 179                           | SetBooleanProperty method            |     |
| RETRY_TIMEOUT variable                                 |                               | iAnywhere.QAnywhere.Client namespace | 232 |
| QAnywhere C++ API                                      | 125                           | setBooleanStoreProperty function     |     |
| rollback function                                      |                               | QAnywhere C++ API                    | 148 |
| QAnywhere C++ API                                      | 172                           | SetBooleanStoreProperty method       |     |
| Rollback method                                        |                               | iAnywhere.QAnywhere.Client namespace | 213 |
| iAnywhere.QAnywhere.Client namespace                   | 246                           | setByteProperty function             |     |
| rule variables                                         |                               | QAnywhere C++ API                    | 163 |
| QAnywhere transmission rules                           | 110                           | setByteStoreProperty function        |     |
| rules                                                  | <i>see</i> transmission rules | QAnywhere C++ API                    | 148 |
| rules file                                             |                               | setDoubleProperty function           |     |
| about QAnywhere                                        | 102                           | QAnywhere C++ API                    | 163 |
| QAnywhere Agent -policy option                         | 89                            | SetDoubleProperty method             |     |
| running MobiLink with messaging and a JMS Connector    |                               | iAnywhere.QAnywhere.Client namespace | 233 |
| QAnywhere                                              | 43                            | setDoubleStoreProperty function      |     |
| running MobiLink with simple messaging                 |                               | QAnywhere C++ API                    | 149 |
| QAnywhere                                              | 33                            | SetDoubleStoreProperty method        |     |
| running the QAnywhere Agent                            |                               | iAnywhere.QAnywhere.Client namespace | 213 |
| about                                                  | 37                            | setFloatProperty function            |     |
| <b>S</b>                                               |                               | QAnywhere C++ API                    | 163 |
| scenario for messaging with external messaging systems |                               | SetFloatProperty method              |     |
| QAnywhere                                              | 8                             | iAnywhere.QAnywhere.Client namespace | 233 |
| scenario for messaging with push notifications         |                               | setFloatStoreProperty function       |     |
| QAnywhere                                              | 6                             | QAnywhere C++ API                    | 149 |
| schedule syntax                                        |                               | SetFloatStoreProperty method         |     |
| QAnywhere transmission rules                           | 105                           | iAnywhere.QAnywhere.Client namespace | 213 |
| scheduled policy                                       |                               | setInReplyToID function              |     |
| QAnywhere agent                                        | 88                            | QAnywhere C++ API                    | 164 |
| schedules                                              |                               | setIntProperty function              |     |
| QAnywhere transmission rules                           | 105                           | QAnywhere C++ API                    | 164 |
| security                                               |                               | SetIntProperty method                |     |
| QAnywhere                                              | 95                            | iAnywhere.QAnywhere.Client namespace | 234 |
| sending messages                                       |                               | setIntStoreProperty function         |     |
| QAnywhere                                              | 67                            | QAnywhere C++ API                    | 149 |
| transactions in QAnywhere                              | 76                            | SetIntStoreProperty method           |     |
| sending QAnywhere messages                             |                               | iAnywhere.QAnywhere.Client namespace | 214 |
| about                                                  | 67                            | setLongProperty function             |     |
| server message stores                                  |                               | QAnywhere C++ API                    | 164 |
| QAnywhere architecture                                 | 5                             | SetLongProperty method               |     |
| setting up in QAnywhere                                | 32                            | iAnywhere.QAnywhere.Client namespace | 214 |
| setAddress function                                    |                               | setMessageID function                |     |
| QAnywhere C++ API                                      | 162                           | QAnywhere C++ API                    | 164 |
| setBooleanProperty function                            |                               | setMessageListener function          |     |
| QAnywhere C++ API                                      | 163                           |                                      |     |



|                                      |      |                                                  |      |
|--------------------------------------|------|--------------------------------------------------|------|
| QAnywhere C++ API                    | 150  | QAnywhere                                        | 10   |
| setMessageListener method            |      | setting up client-side components                |      |
| iAnywhere.QAnywhere.Client namespace | 215  | QAnywhere                                        | 35   |
| setPriority function                 |      | setting up failover                              |      |
| QAnywhere C++ API                    | 165  | QAnywhere                                        | 53   |
| setProperty function                 |      | setting up QAnywhere messaging                   |      |
| QAnywhere C++ API                    | 150  | about                                            | 31   |
| setProperty method                   |      | setting up server-side components                |      |
| iAnywhere.QAnywhere.Client namespace | 215, | QAnywhere                                        | 32   |
| 234                                  |      | setting up the client message store              |      |
| setRedelivered function              |      | QAnywhere                                        | 35   |
| QAnywhere C++ API                    | 165  | setting up the server message store              |      |
| setReplyToAddress function           |      | QAnywhere                                        | 32   |
| QAnywhere C++ API                    | 165  | shutting down QAnywhere                          |      |
| SetSbyteProperty method              |      | about                                            | 77   |
| iAnywhere.QAnywhere.Client namespace | 235  | -si option                                       |      |
| SetSbyteStoreProperty method         |      | QAnywhere Agent [qaagent]                        | 91   |
| iAnywhere.QAnywhere.Client namespace | 215  | simple messaging                                 |      |
| setShortProperty function            |      | QAnywhere architecture                           | 5    |
| QAnywhere C++ API                    | 165  | simple messaging scenario                        |      |
| setShortProperty method              |      | QAnywhere                                        | 5    |
| iAnywhere.QAnywhere.Client namespace | 235  | SMS                                              |      |
| setShortStoreProperty function       |      | using QAnywhere notifications with               | 40   |
| QAnywhere C++ API                    | 151  | SQL Anywhere Studio                              |      |
| setShortStoreProperty method         |      | documentation                                    | viii |
| iAnywhere.QAnywhere.Client namespace | 216  | start function                                   |      |
| SetStoreProperty method              |      | QAnywhere C++ API                                | 151  |
| iAnywhere.QAnywhere.Client namespace | 216  | Start method                                     |      |
| setStringProperty function           |      | iAnywhere.QAnywhere.Client namespace             | 217  |
| QAnywhere C++ API                    | 166  | starting the MobiLink synchronization server for |      |
| setStringProperty method             |      | QAnywhere messaging                              |      |
| iAnywhere.QAnywhere.Client namespace | 236  | about                                            | 33   |
| setStringStoreProperty function      |      | stop function                                    |      |
| QAnywhere C++ API                    | 151  | QAnywhere C++ API                                | 151  |
| setStringStoreProperty method        |      | Stop method                                      |      |
| iAnywhere.QAnywhere.Client namespace | 217  | iAnywhere.QAnywhere.Client namespace             | 217  |
| setText function                     |      | -su option                                       |      |
| QAnywhere C++ API                    | 170  | QAnywhere Agent [qaagent]                        | 92   |
| setTimestamp function                |      | support                                          |      |
| QAnywhere C++ API                    | 166  | newsgroups                                       | xiii |
| setting properties in a file         |      | synchronous message receipt                      |      |
| QAManager                            | 64   | QAnywhere                                        | 69   |
| setting properties programmatically  |      | system messages                                  |      |
| QAManager                            | 65   | QAnywhere                                        | 73   |
| setting QAManager properties         |      | <b>T</b>                                         |      |
| about                                | 64   | technical support                                |      |
| setting up                           |      |                                                  |      |

newsgroups xiii

TestMessage application

    QAnywhere tutorial 11, 18

    source code 22

Text property

    iAnywhere.QAnywhere.Client namespace 240

TextLength property

    iAnywhere.QAnywhere.Client namespace 240

Timestamp property

    iAnywhere.QAnywhere.Client namespace 227

transactional messaging

    QAnywhere 75

TRANSACTIONAL variable

    QAnywhere C++ API 122

transactions

    QAnywhere messages 75

transmission rule variables

    QAnywhere 110

transmission rules

    about 102

    delete rules 118

    QAnywhere 101

    QAnywhere examples 103

    QAnywhere syntax 105

    variables 110

triggerSendReceive function

    QAnywhere C++ API 151

TriggerSendReceive method

    iAnywhere.QAnywhere.Client namespace 218

tutorials

    QAnywhere 11

**U**

understanding QAnywhere addresses

    about 59

using JMS connectors

    QAnywhere 42

using notifications with SMS

    QAnywhere 40

using push notifications

    QAnywhere 40

using QAnywhere messaging and MobiLink data synchronization together

    about 51

**V**

-v option

QAnywhere Agent [qaagent] 92

variables

    QAnywhere transmission rules 110

**W**

WebLogic

    QAnywhere and 8

    what QAnywhere does 3

writeBinary function

    QAnywhere C++ API 132

WriteBinary method

    iAnywhere.QAnywhere.Client namespace 188

writeBoolean function

    QAnywhere C++ API 132

WriteBoolean method

    iAnywhere.QAnywhere.Client namespace 188

writeByte function

    QAnywhere C++ API 132

writeChar function

    QAnywhere C++ API 132

WriteChar method

    iAnywhere.QAnywhere.Client namespace 189

writeDouble function

    QAnywhere C++ API 133

WriteDouble method

    iAnywhere.QAnywhere.Client namespace 189

writeFloat function

    QAnywhere C++ API 133

WriteFloat method

    iAnywhere.QAnywhere.Client namespace 189

writeInt function

    QAnywhere C++ API 133

WriteInt method

    iAnywhere.QAnywhere.Client namespace 190

writeLong function

    QAnywhere C++ API 133

WriteLong method

    iAnywhere.QAnywhere.Client namespace 190

WriteSbyte method

    iAnywhere.QAnywhere.Client namespace 190

writeShort function

    QAnywhere C++ API 133

WriteShort method

    iAnywhere.QAnywhere.Client namespace 191

writeString function

    QAnywhere C++ API 134

WriteString method

|                                       |     |
|---------------------------------------|-----|
| iAnywhere.QAnywhere.Client namespace  | 191 |
| writeText function                    |     |
| QAnywhere C++ API                     | 170 |
| WriteText method                      |     |
| iAnywhere.QAnywhere.Client namespace  | 241 |
| writing QAnywhere client applications |     |
| about                                 | 55  |
| writing secure messaging applications |     |
| QAnywhere                             | 95  |

## X

|                                         |    |
|-----------------------------------------|----|
| -x option                               |    |
| QAnywhere Agent [qaagent]               | 93 |
| xjms.jndi.authName property             |    |
| QAnywhere                               | 44 |
| xjms.jndi.factory property              |    |
| QAnywhere                               | 44 |
| xjms.jndi.password.e property           |    |
| QAnywhere                               | 45 |
| xjms.jndi.url property                  |    |
| QAnywhere                               | 45 |
| xjms.password.e property                |    |
| QAnywhere                               | 45 |
| xjms.queueConnectionAuthName property   |    |
| QAnywhere                               | 45 |
| xjms.queueConnectionPassword.e property |    |
| QAnywhere                               | 45 |
| xjms.queueFactory property              |    |
| QAnywhere                               | 45 |
| xjms.receiveDestination property        |    |
| QAnywhere                               | 45 |
| xjms.topicConnectionAuthName property   |    |
| QAnywhere                               | 45 |
| xjms.topicConnectionPassword.e property |    |
| QAnywhere                               | 45 |
| xjms.topicFactory property              |    |
| QAnywhere                               | 45 |