



Messaging Services Users Guide for
Adaptive Server® Enterprise

Real-Time Data Services

4.0

DOCUMENT ID: DC00464-01-0400-02

LAST REVISED: December 2007

Copyright © 2004-2007 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. 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 Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book	v	
CHAPTER 1	Introduction	1
	RTDS messaging concepts	1
	Automatic decisions in real time	2
	Messaging models	3
	JMS	3
	WebSphere MQ messaging models	4
	Message format	4
	JMS message properties	5
	MQ message topics	5
	Message selectors	5
CHAPTER 2	Understanding Real-Time Data Services	7
	Sending and receiving messages from a queue	7
	Publishing and consuming messages from a JMS topic	8
	Working with message properties	8
	Previewing the messaging interface	9
	MQ overview	11
	MQ publish/subscribe	13
	Syntax for topics	16
	Publisher and subscriber identities	18
	MQ publish/subscribe examples	18
	Transactional message behavior	25
	Transactional messaging set option	25
	MQ security	26
	Connecting to the MQ queue manager	26
	Installing MQ client on Adaptive Server host machines	27
	Installing MQ client libraries	27
	MQ authorizations	28
CHAPTER 3	SQL Reference	29
	Message-related global variables	30

<msgheader> and <msgproperties> documents	36
Adaptive Server-specific message properties	38
Keywords	39
Stored procedures	40
Built-in functions	40
Syntax segments	41
sp_configure 'enable real time messaging'	42
sp_engine	44
sp_msgadmin	48
msgconsume	59
msgpropcount	62
msgproplist	63
msgpropname	65
msgproptype	66
msgpropvalue	68
msgpublish	70
msgrecv	74
msgsend	90
msgsubscribe	126
msgunsubscribe	128
endpoint	131
option_string	134
sizespec	135
timespec	136
CHAPTER 4	
Samples	139
Sybase directories	139
Using code samples with Replication Server function strings	140
Using code samples with SQL	140
Using code samples with Java/JDBC	140
Glossary	141
Index	143

About This Book

Audience

This book describes how to use Sybase® Real-Time Data Services (RTDS) to provide a way to capture transactions (data changes) in an Adaptive Server® Enterprise database and deliver them as events to external applications in real time. These data changes—or events—are delivered to applications through a Java Messaging Service message bus such as EAServer Java Messaging Service (JMS), TIBCO Enterprise Message System (EMS), or IBM WebSphere MQ.

Note Real-Time Data Service version 4.0 is not certified on Adaptive Server version 15.0.2 or later. If you have RTDS 4.0 and you upgrade Adaptive Server to version 15.0.2, 15.0.2 ESD#1, or 15.0.2 ESD#2, RTDS messaging stops. To use the RTDS feature, upgrade to RTDS 4.5.

How to use this book

This book helps you configure and use real-time messaging in Adaptive Server database applications. It includes these chapters:

- Chapter 1, “Introduction,” discusses messaging concepts, models, and formats, and provides a short glossary of terms.
- Chapter 2, “Understanding Real-Time Data Services,” is an overview of Real-Time Data Services (RTDS) specific to Adaptive Server.
- Chapter 3, “SQL Reference,” documents the SQL stored procedures, functions, and global variables for managing and administering real-time messaging, and the general format of option strings.
- Chapter 4, “Samples,” provides code samples that illustrate messaging functionality.

Reference documents

Real-Time Data Services documentation The following make up the Real-Time Data Services documentation set:

- *Messaging Services User's Guide for Adaptive Server Enterprise* (this book) – explains how to use Real-Time Data Services with Adaptive Server Enterprise.

-
- *Installation and Release Bulletin* – contains installation instructions and last-minute information that was too late to be included in the *Messaging Services User's Guide* (this book).

A more recent version of this installation and release bulletin may be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase Product Manuals Web site. To access the most recent release bulletin:

- a Go to Product Manuals at <http://www.sybase.com/support/manuals/>.
- b Follow the links to the appropriate Sybase product.
- c Select the Release Bulletins link.
- d Select the Sybase product version from the Release Bulletins list.
- e From the list of individual documents, select the link to the release bulletin for your platform. You can either download the PDF version or browse the document online.

Adaptive Server Enterprise documentation The following documents make up the Adaptive Server Enterprise documentation set:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

A more recent version of this installation and release bulletin may be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manuals Web site.

- The installation guide for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What's New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 15.0, the system changes added to support those features, and the changes that may affect your existing applications.
- *ASE Replicator User's Guide* – describes how to use the ASE Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.
- *Component Integration Services User's Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- The configuration guide for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.

- *Enhanced Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server User's Guide* – describes how to use Historical Server to obtain performance information for SQL Server[®] and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install, configure, and use Java classes and methods in Adaptive Server.
- *Job Scheduler User's Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Monitor Client Library Programmer's Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server User's Guide* – describes how to use Monitor Server to obtain performance statistics from Adaptive Server.
- *Performance and Tuning Guide* – is a series of four books that explains how to tune Adaptive Server for maximum performance:
 - *Basics* – the basics for understanding and investigating performance questions in Adaptive Server.
 - *Locking* – how the various locking schemas can be used for improving performance in Adaptive Server.
 - *Monitoring and Analyzing* – how statistics are obtained and used for monitoring and optimizing performance.
 - *Optimizer and Abstract Plans* – how the optimizer processes queries and how abstract plans can be used to change some of the optimizer plans.
- *Query Processor* – describes the Query Processor in Adaptive Server Enterprise and how it is used to optimize query processing in Adaptive Server.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book.

-
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL[®] information:
 - *Building Blocks* – Transact-SQL datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
 - *Commands* – Transact-SQL commands.
 - *Procedures* – Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
 - *Tables* – Transact-SQL system tables and dbcc tables.
 - *System Administration Guide* – consists of two volumes that provide in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.
 - *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
 - *Transact-SQL User's Guide* – documents Transact-SQL, the Sybase-enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the pubs2 and pubs3 sample databases.
 - *Troubleshooting and Error Messages Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
 - *Unified Agent and Agent Management Console User's Guide* – explains how Unified Agent provides runtime services to manage, monitor, and control distributed Sybase resources.
 - *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
 - *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase Failover to configure an Adaptive Server as a companion server in a high availability system.
 - *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.

- *Web Services User's Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.
- *XML Services* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that comprise XML Services.
- *FLEXnet Licensing End User Guide* – describes the FLEXnet network licensing system.
- *SAMreport Users Guide* – explains how to use Software Asset Management (SAM) Solutions, a report generator that monitors the usage of applications that use FLEXlm or SAMwrap.

EAServer documentation The following make up the EAServer documentation set. Use these documents if you are using Real-Time Messaging with EAServer JMS:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

A more recent version of this installation and release bulletin may be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manuals Web site.
- The installation guide for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *Cookbook* – contains tutorials and explains how to use the sample applications included with EAServer.
- *Web Services Toolkit User's Guide* – describes how to create and manage Web services using the various tools, services, and GUIs.
- *Programmer's Guide* – contains information about how to build distributed applications that run on EAServer.
- *Feature Guide* – describes the features of EAServer.
- *System Administration Guide* – contains information about configuring and running EAServer.
- *API Reference Manual* – contains reference pages for EAServer proprietary Java classes, C++ classes, ActiveX interfaces, and C routines.

-
- *Message Bridge for Java User's Guide* – describes how to use Message Bridge for Java
 - *What's New in EAServer* – describes the new features in EAServer.
 - *Security Administration and Programming Guide* – describes the features in EAServer with which you can define the security characteristics of client/server communications.
 - *Performance and Tuning Guide* – contains information about configuring server and application settings to achieve the highest application performance, and also describes implementation and design issues that affect performance.
 - *Troubleshooting Guide* – contains procedures for troubleshooting problems that EAServer users may encounter.

RepConnector documentation The following make up the RepConnector™ documentation set. Use these documents if you are using Real-Time Messaging with RepConnector:

- *Release Bulletin for UNIX and Windows* – contains last-minute information that was too late to be included in the books

A more recent version of this installation and release bulletin may be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manuals Web site.
- *Installation Guide for Microsoft Windows and UNIX* – describes installation, upgrade, and configuration procedures for RepConnector
- *Configuration and User's Guide* – provides instructions for performing specific configuration tasks for RepConnector

Replication Server documentation The following make up the Replication Server™ documentation set. Use these documents if you are using Real-Time Messaging with Replication Server:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

A more recent version of this installation and release bulletin may be available on the Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manuals Web site.
- The installation guide for your platform – describes installation and upgrade procedures for all Replication Server and related products.

- *What's New in Replication Server?* – describes the new features in Replication Server version 15.0 and the system changes added to support those features.
- *Administration Guide* – contains an introduction to replication systems. This manual includes information and guidelines for creating and managing a replication system, setting up security, recovering from system failures, and improving performance.
- *Configuration Guide* for your platform – describes configuration procedures for all Replication Server and related products, and explains how to use the `rs_init` configuration utility.
- *Design Guide* – contains information about designing a replication system and integrating heterogeneous data servers into a replication system.
- *Getting Started with Replication Server* – provides step-by-step instructions for installing and setting up a simple replication system.
- *Heterogeneous Replication Guide* – describes how to use Replication Server to replicate data between databases supplied by different vendors.
- *Reference Manual* – contains the syntax and detailed descriptions of Replication Server commands in the Replication Command Language (RCL); Replication Server system functions; Sybase Adaptive Server commands, system procedures, and stored procedures used with Replication Server; Replication Server executable programs; and Replication Server system tables.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
- *Troubleshooting Guide* – contains information to aid in diagnosing and correcting problems in the replication system.
- Replication Server plug-in help, which contains information about using Sybase Central™ to manage Replication Server.

Related documents

- Java Message Service by Java Technologies at <http://java.sun.com/products/jms>.
- TIBCO Enterprise Message Service by TIBCO Software at <http://www.tibco.com>.
- IBM WebSphere MQ by IBM at <http://www-306.ibm.com/software/integration/wmq/>.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

-
- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
 - The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Sybase Product Manuals Web site, go to Product Manuals at <http://www.sybase.com/support/manuals/>.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ Finding the latest information on product certifications

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Select Products from the navigation bar on the left.
- 3 Select a product name from the product list and click Go.
- 4 Select the Certification Report filter, specify a time frame, and click Go.
- 5 Click a Certification Report title to display the report.

❖ Finding the latest information on component certifications

- 1 Point your Web browser to Availability and Certification Reports at <http://certification.sybase.com/>.

- 2 Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
- 3 Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

- 1 Point your Web browser to the Sybase Support Page at <http://www.sybase.com/support>.
- 2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
- 3 Select a product.
- 4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

- 5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

In the regular text of this document, the names of files and directories appear in *italics*, for example:

- Windows: *%SYBASE%\bin*

- UNIX platforms: *\$\$SYBASE*

Note Substitute your Sybase installation drive and directory for *\$\$SYBASE* in UNIX, and *%SYBASE%* in Windows.

Table 1 details the typographic (font and syntax) conventions as used in this document.

Table 1: Font and syntax conventions for this document

Element	Example
Command names, command option names, database names, datatypes, utility names, utility flags, and other keywords are Helvetica.	dsedit
Variables, or words that stand for values that you fill in, are in <i>italics</i> .	select <i>column_name</i> from <i>table_name</i> where <i>search_conditions</i>
<i>Parentheses</i> must be typed as part of the command.	compute row_aggregate (<i>column_name</i>)
<i>Curly braces</i> indicate that at least one of the enclosed options is required by the command (see comma).	{cheese, sauce}
	Note Do not type the curly braces.
<i>Brackets</i> mean that choosing one or more of the enclosed options is optional.	[anchovies, pineapple, bell_peppers]
	Note Do not type the brackets.
The <i>vertical bar</i> means you may select only one of the options shown.	{cash check credit}
	Note Do not type the curly braces.
The <i>comma</i> means you may choose as many of the options shown as you like; separate multiple choices in a command with commas.	[extra_cheese, avocados, sour_cream]
	Note Do not type the brackets.

Element	Example
<p>An <i>ellipsis</i> (...) means that you can <i>repeat</i> the unit that the ellipsis follows as many times as you like.</p>	<pre>buy <i>thing</i> = price [cash check credit] [, <i>thing</i> = price [cash check credit]]...</pre> <ul style="list-style-type: none"> • You must buy at least one <i>thing</i> (item) and give its price. • You may choose a method of payment: one of the options enclosed in square brackets. • You may choose also to buy additional items: as many of them as you like. For each item you buy, provide its name, its price, and (optionally) a method of payment.
<p>Syntax statements, which display the utility's syntax including all its options, appear as shown here, either in san serif font for flags and options (-v), or italics for user-supplied values (<i>username</i>).</p>	<pre>charset [-Ppassword] [-Sserver] [-Iinterface] <i>sort_order</i> <i>charset</i></pre>
<p>Examples that illustrate computer output appear in Courier, as shown:</p>	<pre>pub_id pub_name city state ----- 0736 New Age Books Boston MA 0877 Binnet & Hardley Washington DC (2 rows affected)</pre>

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.



Introduction

Although this book assumes that you have a basic knowledge of messaging systems in database management, this chapter introduces some basic message concepts and models, and provides a short glossary of terms.

Most of the discussion concerns aspects of messaging that are specific to Adaptive Server. This functionality is referred to in this document as real-time messaging services.

Topic	Page
RTDS messaging concepts	1
Automatic decisions in real time	2
Messaging models	3
Message format	4
Message selectors	5

RTDS messaging concepts

Messaging is the exchange of information by two or more software applications. A message is a self-contained package of information.

Many Adaptive Server customers use messaging and queuing, or publishing and subscription systems in their own application environments. These applications are called message-oriented middleware. Often, the same application combines database operations with messaging operations.

Real-Time Data Services (RTDS), simplifies the development of such applications, using Adaptive Server with TIBCO Enterprise Message Service (EMS), EAServer Java Messaging System (JMS), and IBM WebSphere MQ (MQ).

Note EMS is the TIBCO implementation of a Java messaging service (JMS). Unless specified, JMS refers generically to both TIBCO EMS and EAServer JMS in this documentation.

Messaging systems allow senders and receivers to be detached. A messaging system can be asynchronous, in that an application can send messages without requiring receiving applications to be running.

JMS and MQ are APIs that define the way in which clients communicate with message providers. The message sender and the message receiver both act as clients to the message provider.

Messaging systems are provided by message providers. The messaging provider can implement architecture that centralizes or decentralizes the storage of messages, or that is a hybrid of the two.

RTDS performs messaging operations within SQL statements, using built-in functions.

Real-Time Data Services provide a way to capture transactions (data changes) in an Adaptive Server database and deliver them as events to external applications using either:

- JMS message bus, provided by TIBCO and EAServer
- Message Queue Interface (MQI), provided by WebSphere MQ

Automatic decisions in real time

In managing a database, you must sometimes allow for automated decisions in real time, in response to specific events. Real time means that the database can make decisions regarding events at the same time the events occur, rather than simply queuing the events. An event, such as a change in a record, must be evaluated in conjunction with other changes, and the most efficient response chosen. This means that effective decision-support systems need:

- Low latency, enabling real-time enterprise

- An automated system that describes events and the data relating to them
- A technology to reduce the cost of applications that deliver low latency

These business needs are addressed by Sybase Real-Time Data Services (RTDS) using the TIBCO or EAServer JMS message bus, or IBM WebSphere MQ.

Messaging models

This section describes the messaging models for JMS and MQ.

JMS

JMS defines two messaging models:

- Publish-and-subscribe (topics)
- Point-to-point (queues)

Publish-and-subscribe
(topics)

The publish-and-subscribe (pub/sub) model is a one-to-many model. In this type of messaging model, the application sending the message is called the “message producer,” and the applications receiving the message are called “message consumers.” Message consumers establish subscriptions to register an interest in messages sent to a topic. A topic is the destination of this message model.

There are two types of subscriptions you can establish in this model:

- **Durable** – retains messages for the message consumer even when the message consumer application is not connected. The message provider, rather than Adaptive Server, retains the message.
- **Nondurable** – retains messages only when consumer applications are connected to the message provider.

Point-to-point
(queues)

The point-to-point model is a one-to-one model, in the sense that any message sent, by an application called a “message sender,” can be read only by one receiving application, called a “message receiver.” The destination of a point-to-point message is a queue. A queue may contain more than one active message receiver, but the messaging provider ensures that the message is delivered to only one message receiver.

WebSphere MQ messaging models

All MQ messaging models are point-to-point, that is, messages are always sent to, or received from a queue that is managed by a queue manager.

MQ pub/sub is a publish-and-subscribe model built on MQ queues; the messages are not different types of objects. Interaction with MQ pub/sub uses MQ queues.

All messages are sent to the MQ pub/sub **broker**'s broker command queue. This includes registration of a publisher or subscriber, and control messages such as deleting a message, or requesting an update for a message.

A publisher sends a publication to a stream queue. The MQ pub/sub broker distributes the message to all subscribers that have interest in the message. The publisher describes the message using topics, which are subjects that describe the contents of the message.

Subscribers register interest in messages that are sent to a named stream queue by specifying one or more topics of interest. When such messages are sent to the stream queue, the MQ pub/sub broker copies the message to the local queue that the subscriber specified when the subscriber was registered.

Message format

The message format for both MQ and JMS consists of:

- Message header – contains fixed-size portions and variable-sized portions of information specified by the standard. Most of this information is automatically assigned by the message provider.
- Message body – is the application data that client applications exchange.

JMS defines structured message types, such as stream and map, and unstructured message types, such as text, byte, and object.

In MQ, the message body can contain both text and binary data.

JMS message properties

In TIBCO and EAServer, message properties are user-defined properties that you can include with the message. Message properties have types, and these types define application-specific information that message consumers can use later, to select the messages that interest them. Message property types are Java native types `int`, `float`, or `String` (class).

MQ message topics

The MQ, the pub/sub model allows “topics,” which are the subjects of messages. Topics are included in the message in the rules and formatting (RF) header. Unlike JMS, MQ topics are not name-value pairs—which consist of a name and its accompanying value—but are free-form strings that describe the MQ pub/sub message.

Message selectors

JMS – message selectors for TIBCO and EAServer provide a way for message consumers to filter the message stream and select the messages that interest them. These filters apply criteria that reference message properties and their values. The message selector is a SQL-92 `where` clause.

MQ – message selection uses only the message ID and message correlation ID as message selectors. A message reader can selectively choose to read a particular message by specifying a message ID or message correlation ID.

Understanding Real-Time Data Services

This chapter provides an overview of Real-Time Data Services (RTDS) specific to Adaptive Server, which allows you to use Adaptive Server as a client of the message provider. You can send messages to or retrieve messages from the messaging provider by using Transact-SQL commands.

Topic	Page
Sending and receiving messages from a queue	7
Publishing and consuming messages from a JMS topic	8
Working with message properties	8
Previewing the messaging interface	9
MQ overview	11
MQ publish/subscribe	13
Transactional message behavior	25
MQ security	26
Installing MQ client on Adaptive Server host machines	27
MQ authorizations	28

Sending and receiving messages from a queue

Using the built-in functions `msgsend` and `msgrecv`, Transact-SQL applications can send messages to a queue or read messages from a queue in JMS and MQ.

You can use application logic to construct a message body or payload, or it can contain character or binary data directly from relational tables.

You can construct the values of message properties (header or user properties) from relational data or from application logic, and include the constructed message properties in the message that you are sending.

Messages read from the JMS or MQ queue can be processed by the application logic, or directly inserted into relational tables. To filter out only messages of interest when executing the read operation, specify a message selector.

Message properties in read messages can be individually processed by the application logic. For more information about message properties, see `msgsend` on page 90.

Publishing and consuming messages from a JMS topic

Using the built-in functions `msgpublish` and `msgconsume`, Transact-SQL applications can publish messages to, or consume messages from, a JMS topic.

First, you must register a subscription, using `sp_msgadmin 'register'`. Registering a subscription creates a name that `msgpublish`, `msgconsume`, `msgsubscribe`, and `msgunsubscribe` functions can reference. You can register a subscription as **durable** or **nondurable** , and you can specify a message selector to control the messages that come in, ensuring that only messages of interest are read.

You can use `msgsubscribe` to tell the JMS provider to hold messages until the application logic is ready to process them. Use `msgunsubscribe` to tell the JMS provider that the application is no longer interested in messages on this subscription. Use `msgunsubscribe` to delete durable subscriptions from the JMS provider.

Message properties in read messages can be individually processed by the application logic.

See Chapter 3, “SQL Reference” for syntax, parameter, and usage information for `sp_msgadmin` and functions.

Working with message properties

When a message is read, the message header and user properties can be processed by Transact-SQL application logic, using built-in SQL functions. These functions return:

- The name of the n^{th} property

- The value of a named property
- The type of a named property
- The number of properties
- A list of the properties

These built-in functions allow application logic to make processing decisions during runtime, based on the value of the message properties:

- msgproplist
- msgpropname
- msgpropvalue
- msgproptype
- msgpropcount

Previewing the messaging interface

These examples provide a brief preview of the Transact-SQL messaging interface.

Examples

Example 1 JMS – sends a message to a queue:

```
select msgsend('hello world',
  ('eas_jms:iiop://my_eas:7222?queue=queue.sample'
  message property 'city=Detroit')
```

Example 2 JMS – reads a message from a queue, with and without a filter:

```
select msgrecv('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample')

select msgrecv
  ('eas_jms:iiop://my_eas:7222?queue=queue.sample'
  message selector 'city=''Detroit''')
```

Example 3 JMS – publishes a message to a topic:

```
sp_msgadmin register, subscription, sub1,
  'eas_jms:iiop://my_eas:7222?queue=queue.sample'
select msgpublish
  ('hello world', 'sub1' message property 'city=Boston')
```

Example 4 JMS – consumes a message from a topic:

```
select msgconsume('sub1')
```

Example 5 JMS – illustrates working with properties:

```
select msgconsume('sub1')
declare @pcount integer
declare @curr integer
declare @pname varchar(100)
select @curr=1
select @pcount = msgpropcount()
while(@curr<=@pcount)
begin
    select @pname=msgpropname(@curr)
    select msgproptype(@pname)
    select msgpropvalue(@pname)
    select @curr=@curr+1
end
```

Example 6 MQ – sends a message to a queue:

```
select msgsend('hello world',
    'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=DEFAULT.QUEUE'
    message header 'priority=2')
```

Example 7 MQ – reads a message from a queue:

```
select msgrecv(
    'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=DEFAULT.QUEUE'
    option 'timeout=30ss')
```

Example 8 MQ – registers a publisher and publishes a message about “fish”:

```
select msgsend(NULL,
    'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=SYSTEM.BROKER.CONTROL.QUEUE'
    option 'rfhCommand=registerPublisher'
    message header 'topics=fish'
    + ',streamName=ANIMALS.STREAM')
select msgsend('something about a fish',
    'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=ANIMALS.STREAM'
    message header 'topics=fish')
```

Example 9 MQ – registers a subscriber, reads a message, and processes the message properties:

```
select msgsend(NULL,
    'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=SYSTEM.BROKER.CONTROL.QUEUE')
```

```
option 'rfhCommand=registerSubscriber'  
+ ',topics=fish'  
+ ',streamName=ANIMALS.STREAM'  
+ ',queueName=MY_ANIMALS.QUEUE')  
  
select msgrecv(  
  'ibm_mq:chnl1/tcp/host1(1234)?qmgr=QM,queue=MY_ANIMALS.QUEUE'  
  option 'timeout=30ss')  
  
select msgpropvalue('MPQScompcode', @msgproperties)
```

MQ overview

IBM WebSphere MQ allows different applications to communicate asynchronously through queues across different operating systems, different processors, and different application systems.

WebSphere MQ includes the **Message Queue Interface** (MQI), a common low-level **application program interface** (API). Applications use MQI to read and write messages to the queues.

A **queue manager** is a process that manages a set of objects. These objects include queues, **channels**, and process definitions.

A queue object stores messages that are sent by applications to the queue manager. The following are types of queues:

- A local queue – is owned by the queue manager to which a program is connected.
- A remote queue – is owned by a queue manager other than the queue manager to which a program is connected. A remote queue can be sent messages, but messages cannot be read from it.
- An alias queue – is another name for a local or remote queue.
- A dynamic local queue – is a queue that is created on the fly by an application. It is created from a model queue. The persistence of a dynamic queue is defined by the model queue from which it is created.
- A model queue – is a queue that is used as a template for creating a dynamic local queue.
- A cluster queue – is a queue that is owned by a cluster queue manager.

- A channel – is a logical communication link. Channel types are client (client side of a connection), or server (server side of a connection).

A process definition defines a process that executes when incoming messages cause a trigger event.

A WebSphere MQ message consist of two parts:

- Message header – message control information that contains a fixed-sized portion and a variable-sized portion.
- Message body – application data that contains any type of data (text or binary).

When you use `rfhCommand` to publish a publication, if the message payload returned by `msgrecv` is set to:

- `MQRHRF` – the RF header is included in the message body.
- `MQRHRH` – the RF header is not included.

You can obtain the name-value pairs in the RF header by querying `@@msgproperties`.

If the message body contains characters, code-set conversions are available either through MQ native services, or through user exit handlers. The format of the message body is defined by a field in the message header. MQ does not enumerate all possible message body formats, although some formats are provided in samples. Applications can enter any name of the format. For instance, “MQSTR” contains string data and “MQRHRF” contains topics for MQ pub/sub.

WebSphere MQ message types include:

- Datagram – no reply is expected.
- Request – a reply is expected.
- Reply – reply to a request message.
- Report – contains status information from the queue manager or another application.

When messages are sent, various message header properties can be set, such as expiration, persistence, priority, correlation ID, and reply queue.

Message grouping enables you to organize a group of messages into a logically named group. Within a group, each logical message can further be divided into segments. A group is identified by a name, each logical message within a group is identified by a sequence number (starting with 1), and each segment of a logical message is identified by the offset of the message data with respect to the logical message. Segmented messages are not supported by MQ pub/sub, and an attempt to send a segmented message results in an error.

In a queue, messages appear in the physical order in which they were sent to the queue. This means that messages of different groups may be interspersed, and, within a group, the sequence numbers of the messages may be out of order (the latter can occur if two applications are sending messages with the same group ID and partitioned sequence numbers).

When messages are received, the read mode can be either:

- Destructive – message is removed, or
- Nondestructive – the message is retained. This is known as “browsing,” and allows applications to peruse one or more messages before deciding to remove a particular message from the queue.

Receivers can select particular messages by specifying message header properties such as correlation ID or message ID.

When messages are read—as either destructive or nondestructive—the order in which they are returned can be physical or logical. The order is defined by the queue definition. The queue can be defined as being in priority order or first-in, first-out order.

MQ publish/subscribe

WebSphere MQ publish/subscribe is used on MQ queues that employ a broker process to perform subscription resolution. In its simplest form:

- A publisher is the application that is sending the message.
- A subscriber is the application that is receiving the message.
- The following queues are involved:
 - Control queue – where publishers and subscribers send directives to the pub/sub broker. For instance, subscriber registration and deregistration.

- Stream queue – where the publisher sends its messages directly. The pub/sub broker reads the messages from the stream queue and distributes them to the appropriate subscriber’s queue.
- Subscriber queue – where the subscriber reads its messages directly.

Note More queues can be involved, depending on the type of publications.

- The pub/sub broker responds to MQRFH messages sent to the control queue. These command messages control how the pub/sub broker processes messages that arrive on the stream queue. For instance, a subscriber can register an interest in a particular topic.
- The publisher sends messages directly to the stream queue.
- The pub/sub broker reads messages from the stream queue and determines the subscriber queue to which to copy the message. This depends on topics that the subscribers have registered interest in.
- The subscriber reads messages directly from the subscriber queue.

Subscribers register “subscriptions,” which means it is interested in one or more “topics”.

Example

This example, which shows the MQ pub/sub process, uses these variables:

```
declare @BROKER    varchar(100)
declare @STREAM    varchar(100)
declare @SUBQ      varchar(100)
declare @QM        varchar(100)
select @QM         = 'ibm_mq:chnl1/tcp/host1(9876)?qmgr=QM'
select @BROKER     = 'SYSTEM.BROKER.CONTROL.QUEUE'
select @STREAM     = 'ANIMALS'
select @SUBQ       = 'MY_ANIMALS'
```

- 1 Publisher registers to send publications to ANIMALS with topics on fish:

```
select msgsend(NULL,
               @QM + ',queue=' + @BROKER
               option 'rfhCommand=registerPublisher'
               message header 'topics=fish,streamName=' + @STREAM)
```

- 2 Subscriber registers to receive publications published to ANIMALS with topics on fish. The subscriber receives the publications on MY_ANIMALS:

```
select msgsend(NULL,
```

```
@QM + ',queue=' + @BROKER
option 'rfhCommand=registerSubscriber'
message header 'topics=fish'
           + ',streamName=' + @STREAM
           + ',queueName=' + @SUBQ')
```

- Publisher publishes publication to ANIMALS about fish. The MQ pub/sub broker automatically forwards the publication to MY_ANIMALS:

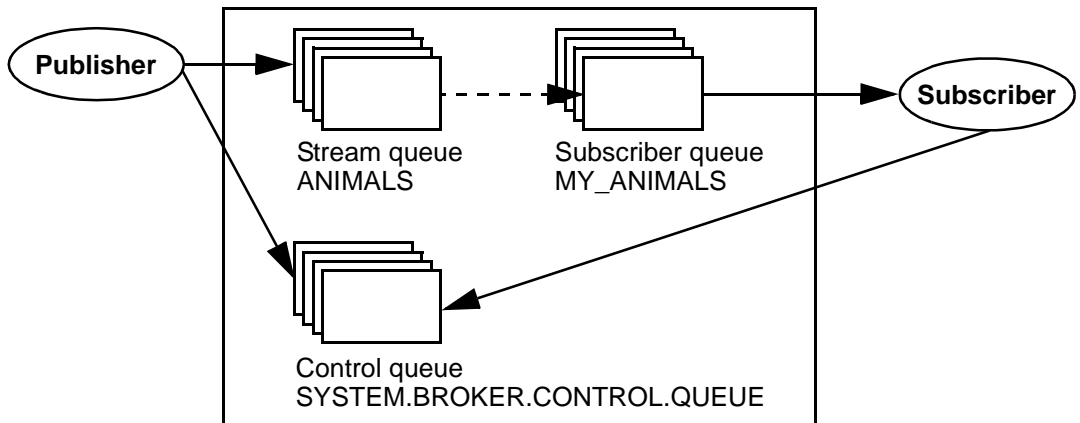
```
select msgsend('something about fish',
              @QM + ',queue=' + @STREAM
              option 'rfhCommand=publish'
              message header 'topics=fish')
```

- Subscriber reads the forwarded message from MY_ANIMALS:

```
select msgrcv(@QM + ',queue=' + @SUBQ option 'timeout=30ss')
```

Figure 2-1 shows the flow of the sample MQ pub/sub process.

Figure 2-1: The MQ publication/subscription process



A message can have one or more topics. WebSphere MQ pub/sub recommends that topics use a hierarchical naming convention as in the examples show below. Subscribers can specify wildcards (such as * and ?) when specifying topics of interest.

These are examples of topics:

```
Sport
```

```
Sport/Soccer
Sport/Tennis
```

These are examples of how subscribers can specify topics of interest:

```
Sport/*           - Any topic about sports.
*/Soccer         - Any topics about soccer.
*/Soccer/Trades  - Any topics about soccer where a 'trade' is involved.
```

A retained publication is a type of publication where the MQ pub/sub broker maintains a copy of a message even after it has delivered it to all subscribers. Normally, a publication is deleted after a copy has been delivered to all subscribers. A retained publication allows a subscriber to asynchronously request the retained publication instead of relying on it being delivered by the MQ pub/sub broker. These types of messages normally contain state information, and are also referred to as state publications.

Syntax for topics

- A topic is generally in the form “topic/subtopic,” for example “sport/baseball.”
- You can specify a wildcard, such as “*” or “?” within a topic.
- When specifying multiple topics, separate the topics with a colon. For instance, “topic1:topic2:topic3:..” and so on.
- If a topic contains spaces or commas, the entire topic list must be placed in quotes. Since topics can appear in message header or message property clauses as strings, if the option string is passed as a quoted scalar value, the enclosed quotes must be escaped by doubling them. Furthermore, if the topic is also contains embedded double quotes, the embedded double quotes must be escaped by quadruple quotes. For example:

```
-- Topic has embedded spaces, we need to quote with escaped quotes
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics=' 'Sport/Football/Hometown Bulldogs''')

-- Topic has embedded spaces, we can quote with double quotes
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics="Sport/Football/Hometown Bulldogs"')
```



```
-- Topic has embedded spaces and embedded double quotes, the inner
-- double quotes need to be escaped.
set quoted_identifier off
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics="quoted ""topic"" here"')

-- Topic has embedded spaces and embedded double quotes, double the
-- quotes around the topic, and quadruple the embedded quotes.
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property "topics=""quoted """"topic"""" here"")
```

- When topics have embedded spaces or quotes, the topic is quoted in the MQRF header. If the topic has embedded quotes, the quotes are escaped before being put into the MQRF header.

In this example, there is one topic, which is placed in the MQRF header as “Sport/Football/Hometown Bulldogs”:

```
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics='Sport/Football/Hometown Bulldogs'')
```

In this example, there is one topic, which is placed in the MQRF header as “Books/”Recipes Of Spain”””””.

```
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics='Books/'Recipes Of Spain'')
```

- You can escape topic name by using “:.”; and any single, non-escaped trailing “:.” is ignored.

In the following example, there are three topics, “baseball”, “baseball/anytown”, and “baseball/scores”.

```
select msgsend(NULL,
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  message property 'topics=baseball:baseball/anytown:baseball/scores')
```

In this example, there are three topics, “subject1”, “subject:2”, and “subject3”. A double-colon (“:”) is used to escape the embedded “:”.

```
select msgsend(NULL,  
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=SAMPLE.STREAM'  
  option 'msgType=datagram,rfhCommand=publish'  
  message property 'topics=subject1:subject::2:subject3')
```

Publisher and subscriber identities

By default, a publisher or subscriber identity consists of:

- Queue name.
- Queue manager name.
- Correlation identifier (optional). You can use the correlation identifier to distinguish between different publishers or subscribers using the same queue. Each publisher and subscriber can be assigned a different correlation identifier. This allows several applications to share a queue. It also allows a single application to differentiate publications originating from different subscriptions.

MQ publish/subscribe examples

Publisher example The Adaptive Server session is a publisher. It publishes on “topicA” and “topicB”; publications on “topicB” are published as retained publications. The retained publication is deleted.

```
-- @QM has the queue manager endpoint  
declare @QM          varchar(100)  
-- @BROKER has the broker queue name  
declare @BROKER      varchar(100)  
-- @STREAM has the stream queue name  
declare @STREAM      varchar(100)  
-- @CORRELID has the generated correlation id  
declare @CORRELID    varchar(100)  
  
-- Put Queue manager name, broker and stream queue names into variables  
select @QM           = 'ibm_mq:chan1/tcp/localhost(5678)?qmgr=QM1'  
select @BROKER       = 'SYSTEM.BROKER.CONTROL.QUEUE'  
select @STREAM       = 'Q1.STREAM'  
  
-- Register the publisher, only for topicA
```

```

select msgsend(NULL, @QM + ',queue=' + @BROKER
               option 'rfhCommand=registerPublisher'
               message header 'correlationAsId=generate'
                               + ',topics=topicA'
                               + ',streamName=' + @STREAM)
-----
0x414d51204652414e4349532e514d202041a3ebfb20014801

-- Save the generated correlation id
select @CORRELID = @@msgcorrelation

-- Send two publications on topicA
select msgsend('topicA, publication 1', @QM + ',queue=' + @STREAM
               option 'rfhCommand=publish'
               message header 'correlationAsId=yes'
                               + ',correlationId=' + @CORRELID
                               + ',topics=topicA')
-----
0x414d51204652414e4349532e514d202041a3ebfb20014803

select msgsend('topicA, publication 2', @QM + ',queue=' + @STREAM
               option 'rfhCommand=publish'
               message header 'correlationAsId=yes'
                               + ',correlationId=' + @CORRELID
                               + ',topics=topicA')
-----
0x414d51204652414e4349532e514d202041a3ebfb20014805

-- Add another topic for this publisher
select msgsend(NULL, @QM + ',queue=' + @BROKER
               option 'rfhCommand=registerPublisher'
               message header 'correlationAsId=yes'
                               + ',correlationId=' + @CORRELID
                               + ',topics=topicB'
                               + ',streamName=' + @STREAM)
-----
0x414d51204652414e4349532e514d202041a3ebfb20014807

-- Publish a retained message on topicB
select msgsend('topicB, retained publication 1', @QM + ',queue=' + @STREAM
               option 'rfhCommand=publish'
               message header 'correlationAsId=yes'
                               + ',correlationId=' + @CORRELID
                               + ',topics=topicB'
                               + ',retainPub=yes')
-----

```

```
0x414d51204652414e4349532e514d202041a3ebfb20014809

-- Publish a second retained publication on topicB
-- This one will replace the current retained publication on topicB.
select msgsend('topicB, retained publication 2', @QM + ',queue=' + @STREAM
    option 'rfhCommand=publish'
    message header ',correlationAsId=Yes'
                + ',correlationId' + @CORRELID
                + ',topics=topicB'
                + ',retainPub=yes')
-----
0x414d51204652414e4349532e514d202041a3ebfb2001480b

-- Delete the retained publication on topicB
select msgsend(NULL, @QM + ',queue=' + @STREAM
    option 'rfhCommand=deletePublication'
    message header 'topics=topicB'
                + ',streamName=' + @STREAM)
-----
0x414d51204652414e4349532e514d202041a3ebfb2001480d

-- Deregister the publisher, for all topics.
select msgsend(NULL, @QM + ',queue=' + @BROKER
    option 'rfhCommand=deregisterPublisher'
    message header 'correlationAsId=yes'
                + ',correlationId=' + @CORRELID
                + ',deregAll=yes'
                + ',streamName=' + @STREAM)
-----
0x414d51204652414e4349532e514d202041a3ebfb2001480f
```

Subscriber example

In this example, the Adaptive Server session subscribes to “topicA” and “topicB”; publications on “topicB” are published as retained publications. This subscriber processes retained publications by requesting an update from the pub/sub broker.

```
-- @QM has the queue manager endpoint
declare @QM          varchar(100)
-- @BROKER has the broker queue name
declare @BROKER     varchar(100)
-- @SUBQUEUE has the subscriber queue name
declare @SUBQUEUE   varchar(100)
-- @STREAM has the stream queue name
declare @STREAM     varchar(100)
-- @CORRELID has the generated correlation id
declare @CORRELID   varchar(100)
```

```

-- Put broker and subscriber queue names into variables
select @QM          = 'ibm_mq:chan1/tcp/localhost(5678)?qmgr=QM1'
select @BROKER      = 'SYSTEM.BROKER.CONTROL.QUEUE'
select @SUBQUEUE    = 'Q1.SUBSCRIBER'
select @STREAM      = 'Q1.STREAM'

-- Register the subscriber, only for topicA
select msgsend(NULL, @QM + ',queue=' + @BROKER
    option 'rfhCommand=registerSubscriber'
    message header 'correlationAsId=generate'
                + ',topics=topicA'
                + ',streamName=' + @STREAM
                + ',queueName=' + @SUBQUEUE)
-----
0x414d51204652414e4349532e514d202041a3ebfb20014801

-- Save the generated correlation id
select @CORRELID = @msgcorrelation

-- Add another topic for this subscriber
-- we will explicitly request update for publications on this topic.
select msgsend(NULL, @QM + ',queue=' + @BROKER
    option 'rfhCommand=registerSubscriber'
    message header 'CorrelationAsId=yes'
                + ',correlationId=' + @CORRELID
                + ',topics=topicB'
                + ',streamName=' + @STREAM
                + ',queueName=' + @SUBQUEUE
                + ',pubOnReqOnly=yes')
-----
0x414d51204652414e4349532e514d202041a3ebfb20014803

-- The publisher now publishes messages in the following order:
-- topicA, topicB (*), topicA, topicB (*)
-- ( '*' denotes a retained publication )

-- Get the first message on the subscriber queue, it will be on topicA.
select msgrcv(@QM + ',queue=' + @SUBQUEUE option 'timeout=30ss')
-----
publication on topicA

-- Get the second message on the subscriber queue, it will be on topicA.
select msgrcv(@QM + ',queue=' + @SUBQUEUE option 'timeout=30ss')
-----
publication on topicA

```

```
-- Request the broker to now send retained publications on topicB
select msgsend(NULL, @QM + ',queue=' + @BROKER
              option 'rfhCommand=requestUpdate'
              message header 'CorrelationAsId=yes'
                             + ',correlationId=' + @CORRELID
                             + ',topics=topicB'
                             + ',streamName=' + @STREAM
                             + ',queueName=' + @SUBQUEUE)
-----
0x414d51204652414e4349532e514d202041a3ebfb20014805

-- Get the next message on the subscriber queue, it will be on topicB.
select msgrecv(@QM + ',queue=' + @SUBQUEUE option 'timeout=30ss')
-----
publication on topicB

-- Get the next message on the subscriber queue, it will be on topicB.
select msgrecv(@QM + ',queue=' + @SUBQUEUE option 'timeout=30ss')
-----
publication on topicB

-- Deregister the subscriber, for all topics.
select msgsend(NULL, @QM + ',queue=' + @BROKER
              option 'rfhCommand=deregisterSubscriber'
              message header 'CorrelationAsId=yes'
                             + ',correlationId=' + @CORRELID
                             + ',deregAll=yes'
                             + ',streamName=' + @STREAM
                             + ',queueName=' + @SUBQUEUE)
-----
0x414d51204652414e4349532e514d202041a3ebfb20014807
```

**Broker response
example**

This example shows how can use request/response messaging to check the response from the pub/sub broker. A subscription is registered by user1, and the pub/sub broker response is checked. The same subscription is then registered again by user2, with a different subscription name, which causes an error response from the pub/sub broker.

Queries executed by user1:

```
-- @QM has the queue manager endpoint
declare @QM          varchar(100)
-- @BROKER has the broker queue name
declare @BROKER      varchar(100)
-- @SUBQUEUE has the subscriber queue name
```

```

declare @SUBQUEUE      varchar(100)
-- @REPLY has the reply queue name
declare @REPLY         varchar(100)

-- Put broker, subscriber and reply queue names into variables
select @QM             = 'ibm_mq:chan1/tcp/localhost(5678)?qmgr=QM1'
select @BROKER        = 'SYSTEM.BROKER.CONTROL.QUEUE'
select @SUBQUEUE     = 'Q1.SUBSCRIBER'
select @REPLY        = 'Q1.REPLY'

-- Register the subscriber.
select msgsend(NULL, @QM + ',queue=' + @BROKER
               option 'rfhCommand=registerSubscriber, msgType=request'
               message header 'correlationAsId=generate'
                           + ',topics=topicA'
                           + ',streamName=Q1.STREAM'
                           + ',queueName=Q1.SUBSCRIBER'
                           + ',replyToQueue=Q1.REPLY')
-----
0x414d51204652414e4349532e514d202041a3ebfb20014801

-- Read the response
select msgrecv(@QM + ',queue=' + @REPLY option 'timeout=30ss')
-----
NULL

-- Check @@msgproperties
select @@msgproperties
-----
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<msgproperties
      MQPSReasonText="&apos;MQRC_NONE&apos;"
      MQPSReason="0"
      MQPSCompCode="0">
</msgproperties>

-- Check MQPSCompCode
if (msgpropvalue('MQPSCompCode', @@msgproperties) != "0")
begin
    print "registerSubscriber failed"
end

```

Queries executed by user2:

```

-- @QM has the queue manager endpoint
declare @QM          varchar(100)

```

```
-- @BROKER has the broker queue name
declare @BROKER          varchar(100)
-- @SUBQUEUE has the subscriber queue name
declare @SUBQUEUE       varchar(100)
-- @REPLY has the reply queue name
declare @REPLY          varchar(100)

-- Put broker, subscriber and reply queue names into variables
select @QM=              'ibm_mq:chan1/tcp/localhost(5678)?qmgr=QM1'
select @BROKER=          'SYSTEM.BROKER.CONTROL.QUEUE'
select @SUBQUEUE=       'Q1.SUBSCRIBER'
select @REPLY=          'Q1.REPLY'

-- Register the subscriber
select msgsend(NULL, @QM + ',queue=' + @BROKER
               option 'rfhCommand=registerSubscriber, msgType=request'
               message header 'correlationAsId=generate'
                          + ',topics=topicA'
                          + ',streamName=Q1.STREAM'
                          + ',queueName=Q1.SUBSCRIBER'
                          + ',replyToQueue=Q1.REPLY')

-----
0x414d51204652414e4349532e514d202041a3ebfb20014801

-- Read the response
select msgrecv(@QM + ',queue=' + @REPLY option 'timeout=30ss')
-----
NULL

-- Check @@msgproperties
select @@msgproperties
-----
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<msgproperties
      MQPSUserId='&apos;user2 &apos;'
      MQPSReasonText='&apos;MQRCCF_DUPLICATE_IDENTITY&apos;'
      MQPSReason="3078"
      MQPSCompCode="2"
</msgproperties>

-- Check MQPSCompCode
if (msgpropvalue('MQPSCompCode', @@msgproperties) != "0")
begin
print "registerSubscriber failed"
end
```


Transactional message behavior

By default, all messaging operations—`msgsend`, `msgrecv`, `msgpublish`, `msgconsume`, `msgsubscribe`, and `msgunsubscribe`—roll back if the database transaction rolls back. However, a failed messaging operation using `msgsend` or `msgrecv` does not affect the parent database transaction.

- If a process included in a transaction executes `msgsend` or `msgpublish`, the resulting message is invisible on the message bus until the process commits the transaction. This is unlike executing a SQL update or insert.

A process that executes SQL update and insert commands in a transaction sees the effect of these commands immediately, before they are committed.

- A process executing `msgsend` or `msgpublish` in a transaction to send a message cannot read that message using `msgrecv` or `msgconsume` until it commits the transaction.

Transactional messaging set option

Transactional behavior is controlled by the `set transactional messaging` command, which provides three modes of operation, allowing you to select preferred behavior when you use messaging functions in a transaction:

```
set transactional messaging [ none | simple | full]
```

- *none* – provides that messaging operations and database operations do not affect each other. In this example, `msgsend` is executed and the message is sent to the message bus, whether insert succeeds or fails:

```
begin tran
  msgsend (...)
  insert (...)
rollback
```

- *simple* (the default setting) – causes database operations to affect messaging operations, but messaging operations do not affect the database transaction. In this example, insert is not aborted if `msgsend` fails:

```
begin tran
  insert (...)
  msgsend (...)
commit
```

In this example, `msgsend` is rolled back:

```
begin tran
    insert (...)
    msgsend (...)
rollback
```

- *full* – provides full transactional behavior. In this mode, messaging operations and database operations affect each other. If the messaging operation fails, the transaction rolls back. If database transactions fail, messaging operations roll back.

```
begin tran
    select @message=msgrecv(Q1,...)
    insert t2 values (@message,...)
    select msgsend ( t2.status,...)
commit tran
```

- When transactional messaging is set to *full* or *simple*, uncommitted transactions that send or publish messages cannot be read within the same transaction.

Transact-SQL applications can specify a preferred mode, depending on their application requirements.

Note You cannot use set transactional messaging inside a transaction.

MQ security

This section discusses security and MQ.

Connecting to the MQ queue manager

You cannot specify a user name and password with the MQ endpoint as you can using JMS. All connections to the MQ queue manager are made as the user that the Adaptive Server process is running as. After making the connection to the MQ queue manager, Adaptive Server then attempts to open the queue as the Adaptive Server login that is performing the operation. For this reason, the user must:

- Have a user account on the machine on which MQ queue manager is running. Without such an account, the user must use the `msgsnd` function's `alter_user=yes` option to perform messaging operations. See Table 3-10 on page 99 in the `msgsnd` reference pages for more information on `alter_user=yes`.
- Be an MQ user.
- Have the MQ authorizations listed in Table 2-2 on page 28.

Note The Adaptive Server “`messaging_role`” is still required to execute Real Time Data Services built-in functions.

In addition, the ‘`register, login`’ and ‘`default, login`’ functions of `sp_msgadmin` do not allow you to register Adaptive Server logins, or to create default Adaptive Server logins if the endpoint specified is a queue manager. See `sp_msgadmin` on page 48 for more information.

Installing MQ client on Adaptive Server host machines

You must install the MQ client software on the Adaptive Server host machine.

Installing MQ client libraries

Adaptive Server dynamically loads the MQ client shared libraries. Table 2-1 shows where to install the shared libraries.

Table 2-1: MQ client shared libraries and directories

Platform	Directory	Library name
Solaris 32	<code>/opt/mqm/lib</code>	<code>libmqmcs.so, libmqic.so</code>
Solaris 64	<code>/opt/mqm/lib64</code>	<code>libmqmcs.so, libmqic.so</code>
Linux 32	<code>/opt/mqm/lib</code>	<code>libmqic_r.so</code>
HP 64	<code>/opt/mqm/lib64</code>	<code>libmqic.sl</code>
AIX 64	<code>/usr/mqm/lib64</code>	<code>libmqic_r.a(mqic_r.o)</code>
Windows	<code>c:\Program Files\IBM\WebSphere MQ\bin</code>	<code>MQIC32.DLL</code>

- *HP, Linux, and Solaris* – Adaptive Server loads the library from `/opt/mqm/lib` so you do not need to set your `LD_LIBRARY_PATH` for MQ.
- *IBM* – set `$LIBPATH` to include `/usr/mqm/lib64`
- *Windows* – set `%PATH%` to include the library

You do not need to set the `LD_LIBRARY_PATH` for MQ. Adaptive Server loads the library from `/opt/mqm/lib` or `/usr/mqm/lib`.

MQ authorizations

MQ configuration requires the following UNIX user account and user group (principle/group) authorizations:

Table 2-2: MQ principle/groups and their authorizations

MQ principle/group	MQ authorization
OS login that is running the data server executable	connect, altusr, inq, and setid on queue manager
OS login of Adaptive Server login that is executing any messaging operation	inq on queue
OS login of Adaptive Server login that is executing the messaging read operation	get on queue
OS login of Adaptive Server login that is executing the messaging browse operation	browse on queue
OS login of Adaptive Server login that is executing the messaging send operation	put on queue
OS login of Adaptive Server login dynamic queue specified as the <code>replyToQueue</code>	crt, dlt on queue manager, and get, inq on Model Queue

SQL Reference

This chapter describes global variables, stored procedures, functions, and syntax segments that you can use to manage and administer Real-Time Data Services (RTDS).

Topic	Page
Message-related global variables	30
<msgheader> and <msgproperties> documents	36
Adaptive Server-specific message properties	38
Keywords	39
Stored procedures	40
Built-in functions	40
Syntax segments	41
sp_configure 'enable real time messaging'	42
sp_engine	44
sp_msgadmin	48
msgconsume	59
msgpropcount	62
msgproplist	63
msgpropname	65
msgproptype	66
msgpropvalue	68
msgpublish	70
msgrecv	74
msgsend	90
msgsubscribe	126
msgunsubscribe	128
endpoint	131
option_string	134
sizespec	135
timespec	136

Message-related global variables

These global variables provide application programs with access to message information from the most recent message sent or received.

`@@msgcorrelation`

Contains correlation from last message sent or read.

- MQ – MQ does not verify whether `@@msgcorrelation` consists of printable characters. Application programs should not rely on `@@msgcorrelation` being in the current server character set, and should use `@@msgcorrelation` only as a selector for subsequent messages. If `@@msgcorrelation` is to be returned to the application, convert it to a varbinary datatype.
- JMS – `@@msgcorrelation` contains the correlationId from the the most recent message sent or received.

`@@msgheader`

Contains message header information from the most recent message received. This variable's format is in XML. For details about this format, see “<msgheader> and <msgproperties> documents” on page 36.

Functions that set `@@msgheader` include `msgrecv` and `msgconsume`.

Table 3-1 lists the valid field names for the `@@msgheader` global variable, and their descriptions for MQ. Table 3-2 on page 32 lists `@@msgheader` fields and descriptions for JMS.

Table 3-1: MQ @@msgheader fields and descriptions

Property name	Description
ApplIdentityData	Application data relating to identity.
ApplOriginData	Application data relating to origin.
CodedCharSetId	Numeric-coded character set identifier.
CorrelId	Correlation identifier.
Encoding	Encoding of binary data in the message. Bit mask of flags in the Encoding field.
DecimalEncoding	This is the encoding for decimal numbers in the message payload, and is a synthesized property derived from the Encoding field. If: <ul style="list-style-type: none"> • BigEndian – decimal numbers are big-endian. • LittleEndian – decimal numbers are little-endian. • Undefined – decimal numbers are not defined as either big-endian or little-endian.
Feedback	Feedback status.

Property name	Description
FloatEncoding	This is the encoding for floating point numbers in the payload, and is a synthesized property derived from the Encoding field. If: <ul style="list-style-type: none"> • BigEndian – floating point numbers are big-endian. • LittleEndian – floating point numbers are little-endian. • Undefined – floating point numbers are not defined as either big-endian or little-endian.
Format	Format name of message data, this can be an MQ-defined format name or an application-defined format name.
GroupId	Group identifier
IntegerEncoding	This is the encoding for integers in the payload, and is a synthesized property that is derived from the Encoding field. If: <ul style="list-style-type: none"> • BigEndian – integers are big endian. • LittleEndian – integers are little endian. • Undefined – the endianness of integers is undefined.
LastMsgInGroup	If: <ul style="list-style-type: none"> • true – message is the last message of a group. • false – message is not the last message of a group.
MsgId	Message identifier.
MsgInGroup	If: <ul style="list-style-type: none"> • true – message is part of a group. • false – message is not part of a group.
MsgSeqNumber	Message sequence number
MessageType	Message type in the form of a decimal number, unless: <ul style="list-style-type: none"> • request – the message is a request message. • reply – the message is a reply message. • datagram – the message is a datagram message. • report – the message is a report message.
NegativeActionNotification	This is a synthesized property, derived from the Report field. The receiving application should generate a negative-action notification (NAN) report. <ul style="list-style-type: none"> • yes – receiving application should generate a NAN report message, and send it to the destinations specified in the ReplyToQ and ReplyToQMGR fields. • no – receiving application should not generate a NAN report message.
Persistence	The persistence of the message. <p>If:</p> <ul style="list-style-type: none"> • persistent – the message is a persistent message. • non-persistent – the message is a non-persistent message.

Property name	Description
PositiveActionNotification	This is a synthesized property derived from the Report field. The receiving application should generate a positive-action notification (PAN) report. If: <ul style="list-style-type: none"> • yes – receiving application should generate a PAN report message, and send it to the destinations specified in the ReplyToQ and ReplyToQMgr fields. • no – receiving application should not generate a PAN report message.
PutAppName	Name of application that put the message.
PutAppType	Type of application that put the message.
PutDate	Date when message was put.
PutTime	Time when message was put.
ReplyCorrelationId	A synthesized property, derived from the Report field. Denotes what to use as the correlation ID of the report message. <ul style="list-style-type: none"> • msgId – the correlation ID of the report message should be set to the message ID of the received message. • correlationId – the correlation ID of the report message should be set to the correlation ID of the received message.
ReplyMsgId	A synthesized property, derived from the Report field. Denotes what to use as the message ID of the report message. <ul style="list-style-type: none"> • new – a new message ID should be used as the message ID of the report message. • original – the message ID of the message received should be used as the message ID of the report message.
ReplyToQ	Name of reply queue.
ReplyToQMgr	Name of the reply queue manager.
Report	Report options from the message. This is a bitmap of MQRO * flags.
UserIdentifier	User identifier.

Table 3-2: JMS @@msgheader fields and descriptions

Property name	Description
correlation	Correlation ID from the message
destination	The name of the destination from the message
encoding	The encoding name of the message
messageid	The message ID from the message
mode	Delivery mode of the message. Values: <ul style="list-style-type: none"> • persistent • non-persistent
priority	The message priority
redelivered	The redelivery status from the message
replyto	The replyto name from the message

Property name	Description
timestamp	The message timestamp
ttl	A time-to-live value from the message that indicates how long a message exists
type	The message type

@@msgid

Contains the ID of the most recent message sent or received.

MQ – MQ does not verify that the *@@msgid* consists of printable characters. Application programs should not rely on *@@msgid* being in the current server character set, and should only use *@@msgid* as a selector for subsequent messages. If *@@msgid* is returning to the application, it should be converted to a varbinary datatype.

Functions that set the variable are:

- JMS – msgsend, msgpublish, msgrecv, msgconsume.
- MQ – msgsend, msgrecv.

@@msgproperties

Contains message properties information from the most recent message received. This variable's format is in XML. For details about this format, see “<msgheader> and <msgproperties> documents” on page 36.

- JMS – the *@@msgproperties* are the user properties from the message.
- MQ – if:
 - The message contains one or more MQRF headers, the name-value pairs in the MQRF headers and inserted into *@@msgproperties*.
 - Since the name-value pairs in the MQRF header can have non-unique names, the names are made unique by appending a “_ddd”, where ddd is an integer extension for uniqueness.

For instance, a MQRF header with these topics:

```
MQPSTopic */baseball
MQPSTopic */baseball/world series
MQPSTopic */sports
```

Results in these properties in *@@msgproperties*:

```
MQPSTopic */baseball
MQPSTopic_1 */baseball/world series
MPQSTopic_2 */sports
```

Functions that set the variable are:

- JMS – msgrecv, msgconsume.
- MQ – msgrecv.

The list below lists RFH name-value pairs that are extracted from the RF header if they are present.

MQPSCommand	MQPSPubOpts	MQPSSStreamName
MQPSCompCode	MQPSPubTime	MQPSSStringData
MQPSCorrelId	MQPSQMgrName	MQPSSSubIdentity
MQPSDelOpts	MQPSQName	MQPSSSubName
MQPSErrorId	MQPSReason	MQPSSSubUserData
MQPSErrorPos	MQPSReasonText	MQPSSSubUserData
MQPSIntData	MQPSRegOpts	MQPSTopic
MQPSParmId	MQPSSeqNum	MQPSUserId

Unrecognized names are ignored. If the value is quoted (“”) in the RF header, the surrounding quotes are removed. In a quoted value, if there are escaped quotes (“”) within the value, doubled quotes are replaced by a single quote.

`@@msgreplyqmgr`

MQ only – contains the ReplyToQmgr name of the last message read.

`@@msgreplytoinfo`

Contains the name (*provider_url*, *queue_name*, *topic_name*, *user_name*) of the topic or queue name used to receive the next message. Can be a permanent or temporary destination.

Functions that set the variable are:

- JMS – msgsend, msgpublish.
- MQ – msgsend.

`@@msgschema`

JMS only – contains the schema of the message or a null value. Contains the value of the Adaptive Server property *ase_message_body_schema*. For more information, see the description of the schema option in msgsend and msgpublish.

Functions that set the variable are: msgsend, msgpublish.

`@@msgstatus`

Contains either the integer error code of the service provider exception, or zero, if the last operation did not raise an exception.

Functions that set the variable are: msgsend, msgpublish, msgrecv, msgconsume.

@@msgstatusinfo

Contains either the error message of the service provider exception, or zero, if the last msgsend, msgpublish, msgrecv, or msgconsume raised an exception, or an empty string.

MQ – contains provider error message of last messaging operation. The MQ client libraries do not provide localized error messages, so you see an error message such as:

```
MQ API call failed with reason code '%s' (%d)
```

The “%s” is substituted with the MQ mnemonic for the MQ reason code.

The “%d” is substituted with the decimal MQ reason code.

Functions that set the variable are:

- JMS – msgsend, msgpublish, msgrecv, msgconsume.
- MQ – msgsend, msgrecv.

@@msgtimestamp

Contains the timestamp included in the message last sent.

Functions that set the variable are: msgsend, msgpublish.

Examples

MQ only – shows request/reply messaging using both @@msgreplytoinfo and @@msgcorrelation:

Session 1 (requester)	Session 2 (receiver)
<pre>select msgsend('sender mmessage', 'ibm_mq:channel1/TCP/host1(5678)' + '?qmgr=QM1' + ',queue=Q100', option 'msgType=request', message property 'correlationId=0x123456' + 'replyToQueue=Q200')</pre>	<pre>select msgrecv('ibm_mq:channel1/TCP/host1(5678)' + '?qmgr=QM1' + ',queue=Q100') select msgsend('receiver reply', @@msgreplytoinfo, option 'msgType=reply' message property 'correlationId=' + @@msgcorrelation)</pre>

Session 1 (requester)	Session 2 (receiver)
<pre>select msgrecv('ibm_mq:channel1/TCP/host1(5678)' + '?qmgr=QM1' + ',queue=Q200' option 'timeout=30ss', + 'correlationID=0x123456')</pre>	

In this example:

- 1 Session 1 sends the request message to Q100, and expects the reply message on Q200. It sets the correlation to 0x123456.
- 2 Session 2 reads a message from Q100, sends a reply message to Q200, and specifies the correlation to 0x123456. The reply queue is obtained from the message that was just read.
- 3 Session 1 reads the reply message from Q200, wanting only message with correlation 0x123456.

Usage

- These global variables are char datatypes, of length 16384.
- You can remove trailing blanks using `rtrim`.
- `@@msgreplytoinfo` contains reply destination information from the message header. It is formatted as an endpoint, as described in `msgsend` on page 90:

JMS only – the password is not included in the value of `@@msgreplytoinfo`. To use this destination as an argument in a subsequent `msgsend` or `msgrecv` call, add:

```
password=<your password>
```

<msgheader> and <msgproperties> documents

Description

The global variables `@@msgheader` and `@@msgproperties` are set with XML `<msgheader>` and `<msgproperties>` documents that contain the header and properties of the returned message. This section specifies the format of those documents.

The general format of a *<msgheader>* and *<msgproperties>* document for properties named PROPERTY_1, PROPERTY_2, and so on has the form described by the DTD templates in the following syntax section.

Syntax	<pre> <!DOCTYPE msgheader [<!ELEMENT msgheader EMPTY> <!ATTLIST property_1 CDATA> <!ATTLIST property_2 CDATA> etc. <!DOCTYPE msgproperties [<!ELEMENT msgproperties EMPTY> <!ATTLIST property_1 CDATA> <!ATTLIST property_2 CDATA> </pre>
--------	---

Examples	<p>These examples show <i><msgheader></i> or <i><msgproperties></i> documents for two select statements:</p>
----------	--

```

select msgsend('Sending message with properties',
              'my_jms_provider?queue=queue.sample',
              message property 'color=red, shape=square')

select msgrecv('my_jms_provider?queue=queue.sample')

select rtrim (@@msgproperties)

<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<msgproperties
  RTMS_MSGBODY_FORMAT='&apos;string&apos;';
  ASE_RTMS_CHARSET='1'
  ASE_RTMS_VERSION='&apos;1.0&apos;';
  ASE_VERSION='&apos;12.5.0.0&apos;';
  shape='&apos;square&apos;';
  color='&apos;red&apos;'; >
</msgproperties>

select rtrim (@@msgheader)

<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<msgheader
  type='&apos;null&apos;';
  timestamp='1080092021000'
  replyto='&apos;queue.sample&apos;';
  redelivered='false'
  priority='4'
  messageid='&apos;ID:E4JMS-SERVER.73018656B39:1&apos;';
  ttl='0'
  destination='&apos;queue.sample&apos;';
  mode='2'
  correlation='&apos;null&apos;';
  encoding='&apos;null&apos;'; >

```

</msgheader>

Usage

- A *<msgheader>* or *<msgproperties>* document for a specified message contains one attribute for each property of the message header or the message properties. The name of the attribute is the name of the property, and the value of the attribute is the string value of the property.
- The values of attributes in *<msgheader>* or *<msgproperties>* documents are replaced with XML entities. *msgpropvalue* and *msgpropname* implicitly replace XML entities with attribute values.
- A *<msgheader>* or *<msgproperties>* document generated by *msgrecv* or *msgconsume* has an XML declaration that specifies the character set of the properties.

Adaptive Server-specific message properties

JMS – to help with debugging, monitoring, and so forth, predefined properties specific to Adaptive Server are included in the properties portion of the JMS message. These properties typically handle messages that either originate from another Adaptive Server, or that may be useful in debugging.

Many of these message properties are included only if you are running *diagserver*, or when certain trace flags are turned on. All properties beginning with “ASE_” are reserved; you cannot set them using *msgsend* or *msgpublish*. Table 3-3 describes these message properties.

Table 3-3: Adaptive Server-specific messages for JMS

Property	Description	When to use
ASE_RTMS_CHARSET	Character set encoding of sent data.	Always
ASE_MSGBODY_SCHEMA	The schema describing the message body or a null value. This schema is non-null only if the user sends the message schema as part of <i>msgsend</i> . If ASE_MSGBODY_FORMAT is <i>xml</i> , this property contains the XML schema describing the payload. This schema is not truncated, even if its value exceeds 16K.	Always
ASE_MSGBODY_FORMAT	The format of the message body: <i>xml</i> , <i>string</i> (in server character set), <i>binary</i> , and <i>unicode</i> (<i>unichar</i> in network order).	Always

Property	Description	When to use
ASE_ORIGIN	Name of the originating Adaptive Server.	Present with diagserver
ASE_RTMS_VERSION	Version of Adaptive Server using RTDS.	Always
ASE_SPID	SPID that sent the message.	Present with diagserver
ASE_TIMESTAMP	The timestamp of Adaptive Server showing the time the message was sent.	Present with diagserver
ASE_VERSION	Version of Adaptive Server that published message.	Always
ASE_VERSIONSTRING	Version string of the Adaptive Server. Provides information about platform, build type, and so on. Useful for debugging.	Present with diagserver

Note These properties are shown for informational purposes only. They may change in the future.

Keywords

Table 3-4 shows the keywords specific to RTDS, and the functions in which these keywords can be legally used.

Table 3-4: Double and triple keywords in RTDS

JMS or MQ	Keywords	Legal commands and functions using keywords
Both	message header	select msgsend(,, message header,,) select msgpublish(,,message header,,)
Both	message property	select msgsend(,, message property,,) select msgpublish(,,message property,,)
JMS	message selector	select msgrecv(,,message selector,,) select msgconsume(,,message selector,,)
JMS	with retain	select msgunsubscribe(,,with retain,,)
JMS	with remove	select msgunsubscribe(,,with remove,,)
Both	transactional messaging none	set transactional messaging none
Both	transactional messaging simple	set transactional messaging simple
Both	transactional messaging full	set transactional messaging full

Stored procedures

The stored procedures you use with RTDS are:

- `sp_configure` 'enable real time messaging' on page 42
- `sp_engine` on page 44
- `sp_msgadmin` on page 48

`sp_msgadmin` and its options do not configure or administer the underlying message provider. For instance, you must still create, delete, and access queues and topics at the messaging provider level.

Note `sp_addexclass` does not accept MQ Q engines for the `anyengine` and `lastonline` parameters.

Built-in functions

The section in this chapter on built-in functions describes the SQL functions for administering Real-Time Messaging, and the general format of option strings. See Table 3-3 on page 38 for Adaptive Server-specific message properties. The SQL functions in this chapter:

- Send and receive messages to queues
- Publish, subscribe, and consume messages relating to message topics
- Handle message properties

The functions listed in this chapter, and their page numbers, are:

- `msgconsume` on page 59
- `msgpropcount` on page 62
- `msgproplist` on page 63
- `msgpropname` on page 65
- `msgproptype` on page 66
- `msgpropvalue` on page 68
- `msgpublish` on page 70

- msgrecv on page 74
- msgsend on page 90
- msgsubscribe on page 126

Syntax segments

The section in this chapter on syntax segments describes the portions of SQL syntax and constraints used in administering Real-Time Messaging.

The syntax segments listed in this chapter, and their page numbers, are:

- endpoint on page 131
- option_string on page 134
- sizespec on page 135
- timespec on page 136

sp_configure 'enable real time messaging'

Description	Enables, disables, or displays current real time messaging configuration.
Syntax	<code>sp_configure "enable real time messaging" , [<i>enable_or_disable</i>], [<i>ibm_mq</i> <i>eas_jms</i> <i>tibco_jms</i>]</code>
Parameters	<p><i>enable_or_disable</i> specifies whether or not to enable or disable the "real time messaging" option. Valid values are:</p> <ul style="list-style-type: none">• 1 – enables real-time messaging.• 0 – disables real-time messaging. <p>If omitted, the current "real time messaging" configuration is returned.</p> <p><i>ibm_mq</i> enables "real time messaging" for IBM MQ only. This option is ignored if <i>enable_or_disable</i> is 0 (zero).</p> <p><i>tibco_jms</i> enables "real time messaging" for TIBCO JMS only. This option is ignored if <i>enable_or_disable</i> is 0 (zero).</p>

Examples Enables real time messaging for all providers :

```
sp_configure "enable real time messaging",1
```

Example 1 Enables real-time messaging for MQ only:

```
sp_configure "enable real time messaging",1,ibm_mq
```

Note Using this stored procedure overwrites your previous setting. For example, if you had previously set real-time messaging to JMS, running this stored procedure disables your JMS setting as it turns on MQ.

Example 2 Enables real-time messaging for TIBCO only:

```
sp_configure "enable real time messaging",1,tibco_jms
```

Example 3 Disables real-time messaging:

```
sp_configure "enable real time messaging",0
```

Usage The *enable_or_disable* parameter works only if the following are installed and set up correctly:

- The appropriate LD_LIBRARY_PATH for your platform
- The provider DLL libraries

- SYBASE licenses
- The SYBASE interface libraries from the CD

See the *Real-Time Data Services Installation and Release Bulletin* for details on paths and file names.

sp_engine

Description	Enables you to bring a Q engine online or take it offline.
Syntax	sp_engine "online offline can_offline shutdown q_online q_offline q_can_offline q_shutdown" , [<i>engine_id</i>]
Parameters	<p>can_offline returns information on whether an engine can be brought offline. If the engine cannot be brought offline, you see the spids of the Adaptive Server sessions that prevent the engine from being offline. You cannot use this parameter to specify a Q engine.</p> <p>engine_id the ID of the engine.</p> <p>The type of the engine that you specify must match the command (online, q_online, and so on). For example, you cannot specify a non-Q engine with q_offline, and you cannot specify a Q engine with offline.</p> <p>This parameter is required for offline, q_offline, can_offline, q_can_offline, shutdown, and q_shutdown.</p> <p>This parameter is not required for online, q_online.</p>
	<p>online brings an engine online. The value of sp_configure "max online Q engines" must be greater than the current number of Q engines online. You must use quotes because online is a reserved keyword. You cannot use this parameter to specify a Q engine.</p>
	<p>offline brings an engine offline. You can also use <i>engine_id</i> to specify an engine to bring offline. You cannot use this parameter to specify a Q engine.</p>
	<p>q_can_offline returns information on whether a Q engine can be brought offline. If the engine cannot be brought offline, you see the spids of the Adaptive Server sessions that prevent the engine from being offline. You must use <i>engine_id</i> to specify whether a Q engine can be taken offline.</p>
	<p>q_offline brings a Q engine offline. You must use <i>engine_id</i> to specify an engine to bring offline.</p>
	<p>q_online brings the next Q engine online.</p>

q_shutdown

forces an engine offline. If there are any tasks with an affinity to this engine, they are killed after a five-minute wait. You must use quotes, as shutdown is a reserved keyword. You must use *engine_id* to specify whether the Q engine can shut down.

shutdown

forces an engine offline. If there are any tasks with an affinity to this engine, they are killed after a five-minute wait. You must use quotes, as shutdown is a reserved keyword. You cannot use this to specify a Q engine.

Examples**Example 1** Manually brings a Q engine online:

```
sp_engine 'q_online'
go

(return status=0)

02:00000:00000:2005/06/08 12:52:21.09 kernel Network and device connection limit is
1014.
02:00000:00000:2005/06/08 12:52:21.24 server Initialized Unilib version 7.2.
02:00000:00000:2005/06/08 12:52:21.24 kernel Q engine 2, os pid 20025 online
02:00000:00000:2005/06/08 12:52:21.33 kernel LDAP dynamic libraries successfully
loaded.
02:00000:00000:2005/06/08 12:52:21.38 kernel IBM MQ dynamic libraries successfully
loaded.
```

Example 2 Takes a Q engine offline:

```
1> select engine, status from sysengines
2> go

engine status
-----
0 online
1 online_q
2 online_q
(3 rows affected)

1> sp_engine 'q_offline', 1
2> go

(return status = 0)
00:00000:00000:2005/06/08 12:55:54.25 kernel engine
2, os pid 20025 offline

1> select engine, status from sysengines
2> go

engine status
```

```
-----  
0 online  
1 online_q  
(2 rows affected)
```

Example 3 Checks to see whether you can take a Q engine offline:

```
1> select engine, status from sysengines  
2> go
```

```
engine status  
-----  
0 online  
1 online_q  
(2 rows affected)
```

```
1> sp_engine 'q_can_offline', 1  
2> go
```

```
spid: 13 has outstanding rtms-connection  
connections.
```

Example 4 Shuts down a Q engine:

```
1> select engine, status from sysengines  
2> go
```

```
engine status  
-----  
0 online  
1 online_q  
(2 rows affected)
```

```
1> sp_engine 'q_shutdown', 1  
2> go
```

```
(return status = 0)
```

```
1> select engine, status from sysengines  
2> go
```

```
engine status  
-----  
0 online  
  
(1 row affected)
```

Usage

- online, offline, can_offline, and shutdown affect only non-Q engines. You see an error if you specify a Q engine with these parameters.

- `q_online`, `q_offline`, `q_can_offline`, and `q_shutdown` affect only Q engines. You see an error if you specify a non-Q engine using these parameters.
- You cannot shut down or take engine 0 offline.
- You can determine the status of an engine, and which engines are currently online with the following query:

```
select engine, status from sysengines
      where status = "online"
```
- `online` and `shutdown` are keywords and must be enclosed in quotes.
- You can bring engines online only if `max online Q engines` is greater than the current number of engines with an online status, and if enough CPU is available to support any additional engines.
- An engine offline can fail or might not immediately take effect if there are server processes with an affinity to that engine.

Permissions

You must be a System Administrator to bring engines online or take them offline.

sp_msgadmin

Description	Configures and administers messaging-related information.
Syntax	<pre>sp_msgadmin 'config', ['jvmlogging', <i>logging_level</i> 'jvmpropertyfile', <i>filepath</i> 'jvmlogfile', <i>filepath</i> 'jvmmaxthreads', <i>thread_number</i> 'jvmminthreads', <i>thread_number</i> 'jvmthreadtimeout', <i>thread_timeout</i> 'jvm', <i>jvm_parameter</i>] sp_msgadmin 'default', 'login', <i>provider_name</i>, <i>provider_login</i>, <i>provider_password</i> sp_msgadmin 'help', ['list' 'register' 'default' 'remove'] sp_msgadmin 'list', [] 'login'[, <i>provider_name</i>, [<i>login_name</i>] 'provider' [, <i>provider_name</i>] 'subscription' [, <i>subscription_name</i>]] sp_msgadmin 'register', ['provider', <i>provider_name</i>, <i>provider_class</i>, <i>messaging_provider_URL</i> 'login', <i>provider_name</i>, <i>local_login</i>, <i>provider_login</i>, <i>provider_password</i> [, <i>role_name</i>] 'subscription', <i>subscription_name</i>, <i>endpoint</i>[, <i>selector</i> [, <i>delivery_option</i> [, <i>durable_name</i>, <i>client_id</i>]]]] ['publisher', <i>publisher_name</i>, <i>endpoint</i>, <i>topic</i> [, <i>broker_queue</i>[, <i>request_queue</i>[, <i>options</i>]]]] sp_msgadmin 'remove', [<i>provider</i>, <i>provider_name</i> 'login', <i>provider_name</i>, <i>local_login</i> [, <i>role</i>] 'subscription', <i>subscription_name</i></pre>

Parameters

client_id

is the identification used by the messaging provider to identify the subscription as durable. *client_id* is a character string value. If you specify either *client_id* or *durable_name*, you must also specify the other, and the subscription is a durable subscription. Otherwise, it is a nondurable subscription.

The *client_id* and *durable_name* combination identifies durable subscriptions with the message provider, and must be unique. No two subscriptions can have the same *client_id* and *durable_name*.

client_id uniqueness extends across the messaging provider. JMS allows a particular *client_id* to be connected only once at any given time. For instance, if one application already has a durable subscription using a specified *client_id*, the *client_id* specified by another application cannot be the same if the applications are to be connected at the same time.

A durable subscription exists even when the client is not connected. The messaging provider saves messages that arrive even while the client is not connected.

A nondurable subscription exists only while the client is connected. The messaging provider discards messages that arrive while the client is not connected.

'config'

allows you to specify various configurations for the Java Virtual Machine (JVM).

'default'

specifies a default. In the case of `sp_msgadmin 'list'`, lists the syntax to specify the default login for a specified message provider.

Note You cannot use `sp_msgadmin 'default'`, `'login'` if endpoint is an MQ queue manager.

delivery_option

specifies whether a SQL session can consume messages that it publishes. Valid values are:

- local – the SQL session can consume messages that it publishes.
- nonlocal – the SQL session cannot consume messages that it publishes.
- null – assumes the value is local.

durable_name

is a character string value. See the description of *client_id*.

endpoint

is the topic to which the subscription is addressed. See the description of *endpoint* in *msgsend* on page 90.

help

provides syntax information about this stored procedure or about particular parameters.

'jvmlogging', *logging_level*

allows you to configure your messaging service to display only the trace information in your code that is higher than your configured level.

logging_level specifies the level using the Apache log4j logging system. The values for *logging_level* are:

- 'all' – returns all the trace information in the code
- 'debug' – returns JVM debug information
- 'fatal' – returns JVM fatal information
- 'off' – turns off logging
- 'info' – returns information-level log information
- 'error' – returns only error log information

The default value for *logging_level* is 'info'.

See the Apache log4j Web site at <http://jakarta.apache.org/log4j/docs/> for more information on the log4j logging system.

'jvmpropertyfile', *filepath*

specifies the property file that JVM uses for your configuration.

filepath defines the location of your file path. This can be any valid path for your property file, including the use of environment variables. The default value is *\$SYBASE/\$SYBASE_ASE/lib/rtms.properties*.

'jvmlogfile', *filepath*

defines the path to the log file that JVM uses for your configuration.

The log information for JVM displays on the console and is written to a single log file. Every time your log file reaches its maximum size of 5MB, JVM automatically creates a new log file and appends a new number at the end of the file (such as *XXX.2*, *XXX.3*, and so on).

The default value for *filepath* is *\$SYBASE/\$SYBASE_ASE/rtms.log*.

'jvmmaxthreads', *thread_number*

specifies the maximum number of Java threads you want to run at the same time in the JVM server's thread pool.

thread_number is the number of threads. When using *jvmmaxthreads*, the value of *thread_number* must be higher than the value of *jvmmthreads*. The default value is 10.

'jvmmthreads', *thread_number*

specifies the maximum number of Java threads you want to run at the same time in the JVM server's thread pool. The value of *thread_number* can be 0 or higher, but must be lower than the value of *jvmmthreads*. The default value is 0.

'jvmthreadtimeout', *thread_timeout*

allows a thread to be automatically destroyed after a specified period of inactivity.

thread_timeout is the number of seconds before a thread is destroyed. The default value is 600 (10 minutes).

'jvm', *jvm_parameter*

defines the parameters you pass to Java when you start JVM.

jvm_parameter is the name of any valid Java parameter string. The default value is "-Xmx500m", which is a generic Java flag that specifies that Java start with 500Mb of allocated RAM. For more information on the Java -Xmx flag, see the Java Web site at <http://java.sun.com>.

list

lists syntax information about message providers, logins, or subscriptions.

'login'

lists information about a particular messaging provider login mapping or about all messaging provider logins. When used with:

- register – registers a login mapping.

Note You cannot use `sp_msgadmin 'register', 'login'` if endpoint is an MQ queue manager.

- default – specifies a default login.
- remove – removes the mapping previously created between an Adaptive Server login and a service provider login, defined by this call:

```
sp_msgadmin 'register', 'login', local_login,...
```

login_name

is a login name.

local_login

is an Adaptive Server login that maps to the local login.

messaging_provider_URL

is the URL of the messaging provider you are registering.

provider

specifies the message provider. When used with:

- register – registers a message provider.
- list – lists information about a particular messaging provider or about all message providers.
- remove – removes a messaging provider previously defined by this call:

```
sp_msgadmin 'register', 'provider', provider_name
```

provider_class

is the class of the messaging provider you are adding. Valid values are:

- EAS_JMS
- TIBCO_JMS
- IBM_MQ

provider_name

is an alias referring to the messaging provider you are adding, which can be as many as 30 characters in length. In the case of `sp_msgadmin 'register', 'provider', provider_name` is an alias for *messaging_provider*. In the case of `sp_msgadmin 'register', 'login', provider_name` is the name of a previously registered provider.

provider_login

is the login name of the messaging provider that *local_login* maps to when connecting to the message provider. It is also the login the provider uses as the default login when sending or receiving messages from the messaging provider specified by *provider_name* when using `sp_msgadmin 'default'`.

provider_password

is the messaging provider password of the *provider_login*.

register

provides stored procedure syntax to register a message provider, login, or subscription.

Note You cannot use `sp_msgadmin 'register', 'login'` or `sp_msgadmin 'register', 'subscription'` if endpoint is an MQ queue manager.

remove

lists the stored procedure syntax to remove a message provider, login, or subscription.

role_name

is a SQL role name. If you specify a *role_name*, the *local_login* is ignored, and the *provider_login* and *provider_password* apply to the *role_name*.

selector

is a message filter that allows a client to select messages of interest. See the description of filters in `msgrecv` on page 74.

subscription

lists information about a particular subscription or about all subscriptions. Specifies the message provider. When used with:

- `register` – registers a subscription.

Note You cannot use `sp_msgadmin 'register' 'subscription'` if the endpoint is an MQ queue manager.

- `list` – lists information about a particular subscription or about all subscriptions.
- `remove` – removes a subscription previously created by:

```
sp_msgadmin 'register' 'subscription', subscription_name, ...
```

subscription_name

is a subscription name.

Examples

Example 1 JMS – logs the level of JVM:

```
sp_msgadmin 'config', 'jvmlogging', 'info'
```

Example 2 JMS – specifies `/usr/1.prop` as the properties file:

```
sp_msgadmin 'config', 'jvmpropertyfile', '/usr/1.prop'
```

Example 3 JMS – defines the log file path as `$$SYBASE/$SYBASE_ASE/rtms.log`:

```
sp_msgadmin 'config', 'jvmlogfile', '$SYBASE/$SYBASE_ASE/rtms.log'
```

Example 4 JMS – specifies the maximum number of threads in the JVM server's thread pool as 100:

```
sp_msgadmin 'config', 'jvmmaxthreads', 100
```

Example 5 JMS – specifies the minimum number of threads in the JVM server's thread pool as 0:

```
sp_msgadmin 'config', 'jvmmintthreads', 0
```

Example 6 JMS – specifies 10 minutes as the amount of time that a thread is idle before it is automatically destroyed:

```
sp_msgadmin 'config', 'jvmthreadtimeout', 600
```

Example 7 JMS – starts the JVM with 500Mb of RAM by using the `-Xmx500m` flag:

```
sp_msgadmin 'config', 'jvm', '-Xmx500m'
```

Example 8 MQ – registers the “mq_provider_1” messaging provider, which has a class of `IBM_MQ` and a URL of `chanl1/TCP/host1(5678)`:

```
sp_msgadmin 'register', 'provider', 'mq_provider_1', 'ibm_mq',  
'chanl1/TCP/host1(5678)'
```

Example 9 JMS – registers the “eas_1” message provider, which has a class of `EAS_JMS` and a url of `iiop://localhost:7222`:

```
sp_msgadmin 'register', 'provider',  
'eas_1', 'eas_jms', 'iiop://localhost:7222'
```

Example 10 JMS – specifies the default login that applies to all unmapped Adaptive Server logins, when using a specified messaging provider for either sending or receiving:

```
sp_msgadmin 'default', 'login', 'my_eas', 'eas_user', 'eas_password'
```

Note You must first register the *provider_name* by calling `sp_msgadmin 'register', 'provider'`.

Example 11 JMS – specifies the default login:

```
sp_msgadmin 'default', 'login', 'one_jms_provider', 'loginsa',  
'abcdef123456'
```

Example 12 JMS – describes the syntax for `sp_msgadmin 'list'`:

```
sp_msgadmin 'help', 'list'
```

Example 13 JMS – checks the default login:

```
sp_msgadmin 'list', 'login', 'my_jms_provider'
```

Example 14 JMS – lists the details for the user with a login of “loginsa”:

```
sp_msgadmin 'list', 'login', 'my_jms_provider', 'loginsa'
```

Example 15 JMS – lists the details for the “my_jms_provider” message provider:

```
sp_msgadmin 'list', 'provider', 'my_jms_provider'
```

Example 16 JMS – lists the details for subscription “subscription_1”:

```
sp_msgadmin 'list', 'subscription', 'subscription_1'
```

Example 17 JMS – registers the login “ase_login1” using messaging provider login “jms_user1” and messaging provider name “my_jms_provider”:

```
sp_msgadmin 'register', 'login', 'my_jms_provider', 'ase_login1',  
  'jms_user1', 'jms_user1_password'
```

Example 18 JMS – registers a login using the messaging provider login “jms_user1” and a specified password used for all unmapped Adaptive Server logins:

```
sp_msgadmin 'register', 'login', 'my_jms_provider', null, 'jms_user1',  
  'jms_user1_password'
```

Example 19 JMS – registers a login with the messaging provider login “jms_user1” and a specified password used for all Adaptive Server logins that have `sa_role` permissions:

```
sp_msgadmin 'register', 'login', 'my_jms_provider', null, 'jms_user1',  
  'jms_user1_password', 'sa_role'
```

Example 20 JMS – registers the “my_jms_provider” messaging provider, which has a class of `TIBCO_JMS` and an IP of `10.23.233.32:4823` as its address:

```
sp_msgadmin 'register', 'provider', 'my_jms_provider', 'TIBCO_JMS',  
  'tcp://10.23.233.32:4823'
```

Example 21 JMS – registers a durable subscription named “durable_sub1”, then `sp_msgadmin 'list'` displays information about the new subscription.

```
sp_msgadmin 'register', 'subscription', 'durable_sub1',  
            'my_jms_provider?topic=topic.sample', null, null, 'durable1', 'client1'  
sp_msgadmin 'list', 'subscription', 'durable_sub1'
```

Example 22 JMS – registers “subscription_1”, a nondurable subscription.

```
sp_msgadmin 'register', 'subscription', 'subscription_1',  
            'my_jms_provider?topic=topic.sample'
```

Note You must first use `sp_msgadmin register, provider` to register “my_jms_provider”.

Example 23 JMS – removes the default login:

```
sp_msgadmin 'remove', 'login', 'my_jms_provider'
```

Example 24 JMS – removes the messaging provider “my_jms_provider”:

```
sp_msgadmin 'remove', 'provider', 'my_jms_provider'
```

Example 25 JMS – removes the Adaptive Server login “ase_login1” associated with the messaging provider “my_jms_provider”:

```
sp_msgadmin 'remove', 'login', 'my_jms_provider', 'ase_login1'
```

Example 26 JMS – removes the default login, indicated by a null login parameter:

```
sp_msgadmin 'remove', 'login', 'my_jms_provider', null
```

Example 27 JMS – removes all logins for role `sa_role` on “my_jms_provider”:

```
sp_msgadmin 'remove', 'login', 'my_jms_provider', null, 'sa_role'
```

Example 28 JMS – removes “subscription_1”:

```
sp_msgadmin 'remove', 'subscription', 'subscription_1'
```

Usage

You cannot use `sp_msgadmin` inside a transaction.

sp_msgadmin 'register'

- When a login name is used to connect to the message provider, login names are resolved in the following order:
 - a Explicit login names and passwords, specified in the endpoint, if provided.
 - b Explicit login mapping for the current Adaptive Server login.

- c The default login name and password for the message provider, and the role corresponding to the Adaptive Server login.
- d The default login name and password for the message provider, with no specific role association.
- e Null login name and password if none of the above apply.
- You can modify the login mapping between the Adaptive Server login and the messaging provider login only by removing and reregistering it with a different set of mappings.
- MQ only – if you enter an endpoint using a registered provider, using `msgsubscribe`, `msgunsubscribe`, `msgpublish`, and `msgconsume` return errors.
- See `sp_msgadmin` on page 48 for usage common to the variants of `sp_msgadmin`.

sp_msgadmin 'remove'

- Removing a messaging provider does not affect messages that are in transit (that is, messages that are in the process of being sent or received) to this message provider.
- `sp_msgadmin 'remove'` does not affect any current connections to the message provider. This means that if a message provider, login, or default is removed while there is a current connection to the specified message provider, the connection is not affected. However, Sybase does not recommend this practice.
- You must specify *local_login* as null if you specify *role_name*.

sp_msgadmin 'config'

- `sp_msgadmin 'config'` is only available for JMS.
- All the values you specify when you call `sp_msgadmin 'config'` are stored in the `sysattributes` table. To retrieve the values, execute:

```
1> select * from sysattributes where class = 21
```

See *Adaptive Server Enterprise: Tables* for more information on `sysattributes`.

- All the parameters available for `sp_msgadmin 'config'` are dynamically configured except for 'jvm'.

Permissions

You must have `messaging_role` to run the `msgsend` and `msgrecv` functions.

You must have `messaging_role` and `sso_role` permissions to issue:

- `sp_msgadmin 'default'`
- `sp_msgadmim 'register'`
- `sp_msgadmin 'remove'`

Any user can issue:

- `sp_msgadmim 'help'`
- `sp_msgadmin 'list'`

msgconsume

Description EAServer JMS only – provides a SQL interface to consume messages that are published to different topics.

Syntax

```
msgconsume_call ::=
msgconsume (subscription_name, option_and_returns)
  subscription_name:= basic_character_expression
  option_and_returns ::= [option_clause] [returns_clause]
  option_clause ::= [,] option_option_string
  returns_clause ::= [,] returns sql_type
  subscriber_name ::= basic_character_expression
  sql_type ::=
    varchar(integer) | java.lang.String | text
    | varbinary(integer) | image
```

Parameters

basic_character_expression
is a Transact-SQL query expression with datatype of char, varchar, or java.lang.String.

option_string
is the general format of *option_string* is specified in option_string on page 134. The special options to use when consuming a message are described in Table 3-5:

Table 3-5: option and option_string values for msgconsume

<i>option values</i>	<i>option_string values</i>	Default	Description
timeout	timespec between -1, 0 – ($2^{31} - 1$)	-1	By default, msgconsume is a blocking command, which blocks the message until it reads the next message from the message bus. If timeout is not -1, msgconsume returns a null value when the timeout interval lapses without reading a message. The values are in number of milliseconds. timeout uses the timespec option. See timespec on page 136 for more information.
requeue	<i>string</i>	None	The name of a destination, queue, or topic on which to requeue messages that Adaptive Server cannot process. If you do not specify requeue, and the message cannot be processed, an error message appears. The endpoint specified must be on the same messaging provider as msgconsume and msgrecv.

subscription_name
is the name of the subscription from which you are consuming messages.

returns
specifies the clause that you want returned.

SQL_type

is the datatype used in SQL statements.

If you do not specify a datatype to be returned, the default is varchar(16384). The legal SQL datatypes are:

- varchar(n)
- text
- java.lang.String
- varbinary(n)
- image
- univarchar(n)

Examples

Example 1 Defines a subscription on the client server, before consuming a message:

```
sp_msgadmin 'register', 'subscription', 'subscription_1',  
  'my_jms_provider?topic=topic.sample,user=user1,password=pwd',  
  'Supplier=12345',null,'durable1', 'client1'
```

Before consuming messages from a subscription, the client first subscribes to the subscription:

```
select msgsubscribe('subscription_1')  
declare @mymsg varchar(16384)  
select @mymsg = msgconsume('subscription_1')
```

Example 2 Declares variables and receives a message from the specified subscription:

```
declare @mymsg varchar (16384)  
select @mymsg = msgconsume('subscription_1',  
  option 'timeout=0')
```

Forwards a message:

```
select msgsend  
  (msgconsume('subscription_1'), 'my_jms_provider?queue=queue.sample')
```

Reads a message and returns it as a varbinary:

```
select msgconsume('subscription_1' returns varbinary(500))
```

Usage

- Unrecognized option names result in an error.

Note This behavior changed with Adaptive Server version 12.5.3a, and differs from earlier versions.

- `msgconsume` reads a message from the topic defined by the `end_point` and `message_filter` specified by the `subscription_name`. It returns a null value if there is a timeout or error, or returns the body of the message it reads.
- Adaptive Server handles only messages of types `message`, `text`, or `bytes`. If Adaptive Server encounters a message it cannot process, and `requeue` is not specified, the message is left on the original queue. Subsequent reads encounter the same message, with the same effect. To prevent this behavior, specify `requeue`. When `requeue` is specified, messages that Adaptive Server cannot handle are placed on the queue specified.

The specified endpoint must exist on the same messaging service provider as the endpoint used in `msgconsume`.

- Adaptive Server issues an error message if the messaging provider issues messages of types other than `message`, `text`, or `bytes`, and if `requeue` is not specified.
- Calling `msgconsume` has these results:
 - The value returned is the `message_body` value returned by the message provider, converted to the specified returns type.
 - The values of `@@msgheader` and `@@msgproperties` are set to `<msgheader>` and `<msgproperties>` documents, which contain the properties of the message that is returned by `msgconsume`.
The general format of `<msgheader>` and `<msgproperties>` documents are described in `<msgheader>` and `<msgproperties>` documents. See “Message-related global variables” on page 30.
 - You can extract the values of a specific property from XML documents `<msgheader>` and `<msgproperties>`, and other related functions, with `msgpropvalue`. For more details, see `msgpropvalue`, below.

Permissions

You must have `messaging_role` to run `msgconsume`.

msgpropcount

Description	Extracts and returns the number of properties or attributes in <code>msg_doc</code> from a <code><msgheader></code> and <code><msgproperties></code> document.
Syntax	<pre>msgpropcount_call ::= msgpropcount([msg_doc]) msg_doc ::= basic_character_expression prop_name ::= basic_character_expression</pre>
Parameters	<p><code>msgpropcount_call</code> makes the request to use the <code>msgpropcount</code> function.</p> <p><code>msg_doc</code> is the <code><msgheader></code> or <code><msgproperties></code> XML document in the form of <code>basic_character_expression</code>. If you do not specify <code>msg_doc</code>, <code>msgpropcount</code> uses the current value of <code>@@msgprproperties</code>.</p> <p><code>prop_name</code> is the property name from which you want to extract a value or type in the form of <code>basic_character_expression</code>.</p>
Examples	<p>This example assumes that a call from <code>msgrecv</code> returns a message with a single property named <code>trade_name</code> and value of “Acme Maintenance” (“Quick & Safe”). The value of the <code>@@msgproperties</code> global variable is then:</p>

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
  <msgproperties
    trade_name='Acme Maintenance (&quot;Quick &amp; Safe&quot;)'>
  </msgproperties>
```

The ampersand and the quotation marks surrounding the phrase `Quick & Safe` are replaced with the XML entities `"`; and `&`, as required by XML convention.

Retrieves the number of properties from the last message retrieved:

```
select msgpropcount (@@msgproperties)
```

msgproplist

Description	Extracts and returns from a <code><msgheader></code> and <code><msgproperties></code> document a string in the format of an <i>option_string</i> with all of the property attributes of <code>msg_doc</code> .
Syntax	<pre>msgproplist_call ::= msgproplist([msg_doc] [returns varchar text]) msg_doc ::= basic_character_expression prop_name ::= basic_character_expression</pre>
Parameters	<p><code>msgproplist_call</code> makes the request to use the <code>msgproplist</code> function.</p> <p><code>msg_doc</code> is the <code><msgheader></code> or <code><msgproperties></code> XML document. A <i>basic_character_expression</i>. If <code>msg_doc</code> is not specified, the current value of <code>@@msgprproperties</code> is used.</p> <p><code>prop_name</code> is the property name from which you want to extract a value or type. A <i>basic_character_expression</i>.</p> <p>returns <i>varchar text</i> specifies the format of the returning message.</p>
Examples	<p>This example assumes that a call from <code>msgrecv</code> returns a message with a single property named “trade_name” and value of “Acme Maintenance” (“Quick & Safe”). The value of the <code>@@msgproperties</code> global variable is then:</p> <pre><?xml version='1.0' encoding='UTF-8' standalone='yes' ?> <msgproperties trade_name='Acme Maintenance (&quot;Quick &amp; Safe&quot;)'> </msgproperties></pre> <p>The ampersand and the quotation marks surrounding the phrase <code>Quick & Safe</code> are replaced with the XML entities <code>&quot;</code>; and <code>&amp;</code>, as required by XML convention.</p> <p>Either of these retrieves the list of properties belonging to a message:</p> <pre>select msgproplist select msgproplist(@@msgproperties)</pre>
Usage	<ul style="list-style-type: none"> If the result of the <code>msgproplist</code> call is more than 16K, the result value contains the word “TRUNCATED”. You should specify “RETURNS text” instead, in this case. You must use other <code>msgprop</code> functions to iterate through the property list and obtain the names and values of the properties.

- If you run `msgproplist` without a return length, any output over the default return value (32) is truncated. To avoid this, specify the length of your returns. For example, this statement is truncated:

```
declare @properties varchar(1000)
select @properties = msgproplist(@@msgproperties returns varchar)
```

However, this one is not:

```
declare @properties varchar (1000)
select @properties= msgproplist(@@msgproperties returns varchar(1000))
```


msgpropname

Description	Extracts and returns the property name from a <code><msgheader></code> and <code><msgproperties></code> document. The result is a null value if the value of the integer parameter is less than one or greater than the number of properties in <code>msg_doc</code> .
Syntax	<pre>msgpropname_call ::= msgpropname(integer[,msg_doc],) msg_doc ::= basic_character_expression prop_name ::= basic_character_expression</pre>
Parameters	<p><i>integer</i> is the index of the value.</p> <p><code>msgpropname_call</code> makes the request to use the <code>msgpropname</code> function.</p> <p><i>msg_doc</i> the <code><msgheader></code> or <code><msgproperties></code> XML document. A <i>basic_character_expression</i>. If <i>msg_doc</i> is not specified, the current value of <code>@@msgprproperties</code> is used.</p> <p><i>prop_name</i> the property name from which you want to extract a value or type. A <i>basic_character_expression</i>.</p>
Examples	<p>Example 1 Assumes that a call from <code>msgrecv</code> returns a message with a single property named <code>trade_name</code> and value of “Acme Maintenance” (“Quick & Safe”). The value of the <code>@@msgproperties</code> global variable is then:</p>

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<msgproperties
  trade_name='Acme Maintenance (&quot;Quick &amp; Safe&quot;)'>
</msgproperties>
```

The ampersand and the quotation marks surrounding the phrase `Quick & Safe` are replaced with the XML entities `"`; and `&`; , as required by XML convention.

Example 2 Returns a null value, because the ninth property does not exist:

```
select msgpropname (9, @@msgproperties)
```

msgproptype

Description	Extracts and returns from a <code><msgheader></code> and <code><msgproperties></code> document the message provider's property type for the <code>msg_doc</code> property with a name that equals <code>prop_name</code> . The result is a null value if <code>msg_doc</code> does not have a property with a name is equal to <code>prop_name</code> .
Syntax	<pre>msgproptype_call ::= msgproptype(prop_name [, msg_doc]) msg_doc ::= basic_character_expression prop_name ::= basic_character_expression</pre>
Parameters	<p><code>msgproptype_call</code> makes the request to use the <code>msgproptype</code> function.</p> <p><code>msg_doc</code> is the <code><msgheader></code> or <code><msgproperties></code> XML document. A <code>basic_character_expression</code>. If <code>msg_doc</code> is not specified, the current value of <code>@@msgprproperties</code> is used.</p> <p><code>prop_name</code> is the property name from which you want to extract a value or type. A <code>basic_character_expression</code>.</p>
Examples	A message is sent with two properties, "integer_prop," which is an integer with value 1234, and "string_prop," which is a string with the value "cat":

```
select msgsend('msgproptype example',
  'tibco_jms:tcp://localhost:7222?queue=queue.sample'
  MESSAGE PROPERTY "integer_prop=1234,string_prop='cat'")
go
```

```
-----
ID:E4JMS-SERVER.82CC311EC:1
(1 row affected)
```

The message is then read back:

```
select msgrecv('tibco_jms:tcp://localhost:7222?queue=queue.sample')
go
```

```
-----
msgproptype example
(1 row affected)
```

The `@@msgproperties` global variable is selected to display what the properties were in the message just received:

```
select @@msgproperties
go
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <msgproperties
    string_prop="&apos;cat&apos;"
    ASE_RTMS_CHARSET="1"
    ASE_ORIGIN="&apos;francis_pinot_2&apos;"
    ASE_SPID="15"
    ASE_MSGBODY_FORMAT="&apos;string&apos;"
    ASE_TIMESTAMP="&apos;2005/06/22 15:01:36.91&apos;"
    ASE_MSGBODY_SCHEMA="&apos;NULL&apos;"
    ASE_RTMS_VERSION="&apos;1.0&apos;"
    ASE_VERSION="&apos;12.5.0.0&apos;"
    integer_prop="1234">
  </msgproperties>
```

(1 row affected)

The first msgproptype call asks for the type of the “integer_prop” property, and returns “Integer”:

```
1> select msgproptype('integer_prop')
2> go
```

```
-----
Integer
(1 row affected)
```

The second msgproptype call asks for the type of the “string_prop” property, and returns “String”:

```
1> select msgproptype('string_prop')
2> go
```

```
-----
String
(1 row affected)
```

Usage

- MQ – when you use msgproptype to query one of the following binary fields contained in the MQ message header, the string “Hex” is returned:
 - MsgId
 - CorrelId
 - GroupId
 - Encoding

For example, the following returns “Hex”:

```
select msgproptype ('Encoding', @@msgheader)
```

msgpropvalue

Description Extracts and returns from a `<msgheader>` and `<msgproperties>` document the value for the `msg_doc` property where the name equals `prop_name`. The result is the property value converted to varchar, and is a null value if `msg_doc` does not have a property with name that is equal to `prop_name`.

Syntax

```
msgpropvalue_call ::= msgpropvalue(prop_name [ , msg_doc] )
                    msg_doc ::= basic_character_expression
                    prop_name ::= basic_character_expression
```

Parameters

`msgpropvalue_call`
makes the request to use the msgpropvalue function.

`msg_doc`
is the `<msgheader>` or `<msgproperties>` XML document. A `basic_character_expression`. If `msg_doc` is not specified, the current value of `@@msgprproperties` is used.

`prop_name`
is the property name from which you want to extract a value or type. A `basic_character_expression`.

Examples

Example 1 These examples assume that a call from msgrecv returns a message with a single property named “trade_name” and value of “Acme Maintenance” (“Quick & Safe”). The value of the `@@msgproperties` global variable is then:

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
  <msgproperties
    trade_name='Acme Maintenance (&quot;Quick &amp; Safe&quot;)'>
  </msgproperties>
```

The ampersand and the quotation marks surrounding the phrase `Quick & Safe` are replaced with the XML entities `"` and `&`, as required by XML convention. The following retrieves the message property `trade_name`:

```
select msgpropvalue(@@msgproperties, 'trade_name')
-----
('Quick & Safe') Acme Maintenance
```

This is the original string that is stored in an Transact-SQL variable or column.

Example 2 Returns a null value because the message retrieved does not have a property named “discount”:

```
select msgpropvalue('discount', @@msgproperties)
```

Example 3 Retrieves the value of the eighth property:

```
select msgpropvalue (msgpropname(8, @@msgproperties))
```

msgpublish

Description JMS only – provides a SQL interface to publish messages to topics.

Syntax

```
message_publish_call ::=
    msgpublish(message_body, subscription_name
               [options_and_properties])
    options_and_properties ::=
        [option_clause] [properties_clause]
        [header_clause]
        option_clause ::= [,] option option_string
        header_clause ::= [,] message header
        option_string
        properties_clause ::=
            [,] message property option_string
        message_body ::= scalar_expression |
            (select_for_xml)
```

Parameters

message_body
is the message you are sending. The message body can contain any string of characters, and can be binary data, character data, or SQLX data.

subscription_name
is the name of the subscription to which you are publishing messages.

option_clause
is the general format of the option name and an *option_string*, specified in the section *option_string* on page 134.

The options you can specify for *msgsend* are in Table 3-6 on page 72.

properties_clause
is either an *option_string* or one of the options listed in the following tables. The options described in Table 3-6 on page 72 are set as a property in the message header or message properties, as indicated in the disposition column of the table. The option value is the property value.

Property names are case sensitive.

If you use a property not listed in Table 3-7 on page 72, it is set as a property in the message properties of the message sent.

scalar_expression

If a message is a SQL *scalar_expression*, it can be of any datatype.

If the type option is not specified, the message type is text if the *scalar_expression* evaluates to a character datatype; otherwise, the message type is bytes.

If the datatype of the *scalar_expression* is not character, it is converted to varbinary using the normal SQL rules for implicit conversion. The binary value of the datatype is included in the message according to the byte ordering of the host machine.

select_for_xml

is a select expression that specifies a for xml clause.

header_clause

allows users to specify only header properties You see an error if you enter an unrecognized header property.

If a recognized header property is specified in both the *message property* and the *message header* clauses, the one in the *message header* clause takes precedence.

You get an error when you specify any unrecognized options in the *option_clause*.

All previously recognized header properties are accepted in the *message header* clause.

Examples

To publish messages, you must define a subscription on the server to which the client is connected:

```
sp_msgadmin 'register', 'subscription', 'subscription_1',
  'my_jms_provider?topic=topic.sample,user=user1,password=pwd',
  'Supplier=12345',null, 'durable1', 'client'
```

The client server can then publish a message to a specified subscription:

```
select msgpublish
  ('Sending order', 'subscription_1',
  MESSAGE PROPERTY 'Supplier=12345')
```

Usage

- Unrecognized options are ignored if you use message property. If you use message header for the msgsend or msgpublish functions, you see an error when you specify unrecognized options.
- The *subscription_name* must have been specified in a call to:

```
sp_msgadmin 'register', 'subscription'
```

Do not specify *subscription_name* in a subsequent call to:

```
sp_msgadmin 'remove', 'subscription'
```

- Table 3-6 lists the options you can specify for msgpublish for JMS.

Table 3-6: Values for the msgpublish option_string parameter

Option	Values	Default	Comments
schema	<ul style="list-style-type: none"> • no • yes • "user_schema" 	no	<p>Enter one of these values:</p> <ul style="list-style-type: none"> • <i>user_schema</i> – is a user-supplied schema describing the message_body. • no – indicates that no schema is generated and sent out as part of the message. • yes – indicates that Adaptive Server generates an XML schema for the message. yes is meaningful only in a message_body that uses the select_for_xml parameter. select_for_xml generates a SQLX-formatted representation of the SQL result set. The generated XML schema is a SQLX-formatted schema that describes the result set document. <p>The schema is included in the message as ASE_MSGBODY_SCHEMA property.</p>
type	text or bytes	text	The message type to send.

- Table 3-7 lists the options and values for the *properties_clause* parameter. If you use a property not listed in Table 3-7, it is set as a property in the message properties of the message sent.

Table 3-7: Values for the msgpublish properties_clause parameter

Option	Values	Default	Disposition	Comments
correlation	<i>string</i>	none	header	Client applications set correlation IDs to link messages together. Adaptive Server sets the correlation ID the application specifies.
mode	<ul style="list-style-type: none"> • persistent • non-persistent 	persistent	header	<p>When you enter:</p> <ul style="list-style-type: none"> • persistent – the message is backed by the JMS provider, using stable storage. If the messaging provider crashes before the message can be consumed, the message is lost, unless mode is set to persistent. • non-persistent and the messaging provider crashes – you may lose a message before it reaches the desired destination.
priority	1 to 9	4	header	<p>The behavior of priority is controlled by the underlying message bus. The values mentioned here apply to JMS.</p> <p>Priorities from 1 to 4 are normal; priorities from 5 to 9 are expedited.</p>

Option	Values	Default	Disposition	Comments
replyqueue	A string containing a <i>queue_name</i>	none	header	<p>If the value of <i>queue_name</i> or <i>topic_name</i> is:</p> <ul style="list-style-type: none"> • <i>syb_temp</i> – Adaptive Server creates a temporary destination and sends information related to the newly created temporary destination as a part of the header information. <p>The type of the temporary destination, queue or topic, depends on whether you specify <i>replyqueue</i> or <i>replytopic</i>. Only the option listed last is used.</p> <ul style="list-style-type: none"> • A destination that already exists – Adaptive Server does not create a new destination, using instead the one specified by the user.
replytopic	A string containing a <i>topic_name</i>	none	header	
ttl	$0 - (2^{63} - 1)$	0	header	<p>ttl refers to time-to-live on the messaging bus. Adaptive Server is not affected by this.</p> <p>Expiry information, which is the duration of time during which the message is valid, in milliseconds. For instance, 60 indicates that the life of the message is 60 milliseconds.</p> <p>A value of 0 indicates that the message never expires.</p> <p>ttl uses the timespec option. See timespec on page 136 for more information.</p>

Permissions

You must have *messaging_role* to run *msgpublish*.

msgrecv

Description

Provides a SQL interface to receive messages from different service endpoints, which must be queues.

msgrecv receives a message from the specified *service_provider* and *service_destination*, and returns that message. The value returned is the message body returned by the service provider, converted to the specified return type.

Syntax

```
msgrecv_call ::=
    msgrecv (end_point options_filter_and_returns)
            options_filters_and_return ::=
                [option_clause] [filter_clause] [returns_clause]
                option_clause ::= [,] option option_string
                filter_clause ::= [,] message selector message_filter
                message_filter ::= basic_character_expression
                returns_clause ::= [,] returns sql_type
                end_point ::= basic_character_expression
                sql_type ::=
                    varchar(integer) | java.lang.String | text
                    | varbinary(integer) | image
                message_filter ::= basic_character_expression
```

Parameters

basic_character_expression

is a SQL query expression with a datatype of char, varchar, or java.lang.String.

end_point

is a *basic_character_expression* where the runtime value is a *service_provider_uri*. The destination of a message.

filter_clause

passes a *message_filter* directly to a specified message provider, which determines its use.

message_filter

is a filter parameter and *basic_character_expression*. The filter value is passed directly to the message provider. Its use depends on the message provider. See the Usage section below for a discussion of message filters.

Any *message_filter* specified to msgrecv is ignored if the provider class is "ibm_mq."

msgrecv

receives a message from the specified *service_provider* and *service_destination*, and returns that message. The value returned is the message body returned by the service provider, converted to the specified return type.

option

is a value shown in Table 3-8 on page 78 for MQ, and Table 3-9 on page 86 for JMS.

Note Unrecognized option names result in an error.

option_string

is the general format of the *option_string* as specified in *option_string* on page 134. The options for *msgrecv* are described in Table 3-8 on page 78 for MQ and Table 3-9 on page 86 for JMS.

returns_clause

is the datatype that you want returned.

If you do not specify a *returns_clause*, the default is `varchar(16384)`.

If you specify a *returns_clause* of type `varbinary` or `image`, the data is returned in the byte ordering of the message.

sql_type

The SQL datatype. The legal SQL datatypes are:

- `varchar(n)`
- `text`
- `java.lang.String`
- `varbinary(n)`
- `image`
- `univarchar(n)`

Examples

Example 1 MQ – a message is read from the queue Q1 with a specified timeout. If no messages are available on Q1 before the timeout of 3 seconds, a null value is returned:

```
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'timeout=3ss')
```

Example 2 MQ – a correlationId is specified without a timeout. The call returns when a message matching the correlationId is available on the queue:

```
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'correlationId=x67a12z99')
```

Example 3 MQ – a groupId is specified, as well as allMsgsInGroup, but a timeout is not specified. This call blocks until all the messages for the groupId specified are available on the queue:

```
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'groupId=g7853b77,allMsgsInGroup=yes')
```

Example 4 MQ – these messages already exist on the queue:

AA BB CC DD EE FF GG HH

The first three messages are read in browse mode (AA-CC), and CC is removed. The browse cursor is then set back to the beginning, and three messages are read in browse mode (AA-DD), and DD is removed. The read that removes CC causes CC to not be included when the browse is repositioned at the beginning. Finally, a read is performed with position set to next, which reads and removes AA. When this example completes, the messages AA, CC, and DD will no longer remain on the queue.

```
-- Browse cursor at the beginning, this will return 'AA'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=first')

-- Browse the next message, this will return 'BB'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=next')

-- Browse the next message, this will return 'CC'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=next')

-- Remove the message under the browse cursor, this will return 'CC'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
```

```

option 'inputMode=browse+Qdefault,position=cursor')

-- Reposition browse cursor at the beginning, this will return 'AA'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=first')

-- Browse the next message, this will return 'BB'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=next')

-- Browse the next message, this will return 'DD'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,browse=next')

-- Read the message under the cursor, this will return 'DD'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,position=cursor')

-- Read the next message in queue order, this will return 'AA'
select msgrecv(
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=Q1',
  option 'inputMode=browse+Qdefault,position=next')

```

Example 5 TIBCO JMS – receives a message from the specified *end_point*:

```

select msgrecv
  ('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample,'
  +'user=jms_user1,password=jms_user1_password')

```

Example 6 JMS – receives a message from the specified *end_point*, using the timeout option and specifying a message selector:

```

declare @mymsg varchar (16384)
select @mymsg = msgrecv('my_jms_provider?queue=queue.sample',
  option 'timeout=1000'
  MESSAGE SELECTOR 'correlationID = 'MSG_001''')

```

Example 7 JMS – forwards a message to the specified endpoint:

```

select msgsend(msgrecv('my_jms_provider?queue=queue.sample'),
  'another_jms_provider?queue=queue2')

```

Example 8 JMS – this msgrecv call consumes only messages from queue.sample when the message property “Name” is equal to “John Smith”:

```
select msgrecv('my_jms_provider?queue=queue.sample',
  MESSAGE_SELECTOR 'Name=' 'John Smith''')
```

Example 9 JMS – illustrates how to insert a text message into a table:

```
create table T1(c1 numeric(5,0)identity, m text)
insert into T1
select msgrecv('my_jms_provider?queue=queue.sample',
  RETURNS text)
```

Example 10 JMS – this example reads a message and returns it as a varbinary.

```
select msgrecv('my_jms_provider?queue=queue.sample'
  returns varbinary(500))
```

Usage

MQ – Table 3-8 lists the available *option* and *option_string* values for properties of msgrecv.

Table 3-8: MQ option and option_string values for msgrecv

<i>option</i> values	<i>option_string</i> values	Default	Description
allMsgsInGroup	<ul style="list-style-type: none"> • yes • no 	no	This option is ignored unless you specify groupId. When you specify: <ul style="list-style-type: none"> • yes – all logical messages of a group must be present on the queue before the first message of a group is returned. • no – not all logical messages of a group are required to be present on the queue before returning the first message of a group.
allSegments	<ul style="list-style-type: none"> • yes • no 	no	When you specify: <ul style="list-style-type: none"> • yes – all messages of a segmented message must be present on the queue before the first message segment is returned. • no – not all messages of a segmented message are required to be present before returning the first message segment.

option values	option_string values	Default	Description
browse	<ul style="list-style-type: none"> • next • next+Lock • first • first+Lock • cursor • cursor+Lock • reopen • reopen+Lock • unlock • null 	null	<p>If you set the the browse property to:</p> <ul style="list-style-type: none"> • null – the message is read and removed from the queue. The position option controls which message is read. • anything other than null – the message is read but not removed from the queue. The ordering depends on the default ordering of the queue (first-in, first-out or priority) <p>If you also:</p> <ul style="list-style-type: none"> • Specify msgId, correlationId, groupId, sequenceId or offset – MQ browses or reads the next message that matches to the selection criteria that you specify. • Specify timeout, and a message matching the selection criteria is not found – the return is a null value. • Do not specify timeout – the msgrecv operation blocks until a message appears in the queue that matches the selection criteria.

<i>option values</i>	<i>option_string values</i>	Default	Description
browse (continued)			<p>If you specify the following for browse:</p> <ul style="list-style-type: none"> • next – the next message is returned. • next+Lock – the message is returned, and the message is locked so that other readers cannot remove it. • first – the first message is returned. If you specify browse=first after you issue one or more browse=next options, the browse cursor repositions to the starting position where the queue was opened. • first+Lock – the first message is returned, and the message is locked so that other readers cannot remove it. • cursor – the message under the browse cursor is returned. Do not use browse=cursor without first performing browse=first, browse=first+Lock, browse=next, or browse=next+Lock. Repeating browse=cursor returns the same message. • cursor+Lock – the message under the cursor is returned, and the message is locked so that other readers cannot remove it. • reopen – the browse cursor is closed, reopened, and positioned at the start. For priority queues, if a higher priority message comes in since the last open, that message appears at the start of the queue. • reopen+Lock – the browse cursor is closed, reopened, positioned at the start, and the first message is locked so that other readers cannot remove it. • unlock – the message under the cursor is unlocked and returned.

<i>option values</i>	<i>option_string values</i>	Default	Description
bufferLength	sizespec 0 or 1 – value		<p>bufferLength-sized buffer is used to read the message.</p> <ul style="list-style-type: none"> The messaging built-in function attempts to allocate a buffer of this length. The command fails if there is not enough memory to allocate the buffer. When you specify msgrecv to return text or image, msgrecv assumes that the message size is the largest message that the specified queue can accommodate, and uses the maxMsgLength queue property. Increase messaging memory if you set maxMsgLength at: <ul style="list-style-type: none"> Its default of 4MB, or A value that is much larger than the actual length of the messages. <p>Sybase recommends you set the maxMsgLength queue property to the minimum allowed for the application so Adaptive Server can use the least amount of memory to read the message. To set maxMsgLength, use the MQ commands (MQSC) tool to change the MAXMSGL attribute on the queue.</p> <p>Defaults bufferLength defaults to either the:</p> <ul style="list-style-type: none"> Minimum of the maxMsgLength that is defined for the queue manager and the target queue, or The length of the return type if it is not text, image or java.lang.String. <p>0 indicates to use the default.</p> <p>For pub/sub messages, bufferLength must include the length of the message topics, including the MQRF header.</p>
closeAfterRecv	<ul style="list-style-type: none"> yes no 	no	<p>If:</p> <ul style="list-style-type: none"> yes – the queue closes after the current msgrecv operation, allowing the queue to be reopened with a different input mode on subsequent msgrecv calls. no – the queue remains open after the current msgrecv operation.
completeMsg	<ul style="list-style-type: none"> yes no 	yes	<p>If:</p> <ul style="list-style-type: none"> yes – segmented messages are returned as a single message. no – if there are segmented messages, each segment is returned as a separate message. <p>completeMsg should have the same setting for all calls to msgrecv for the same endpoint.</p>

option values	option_string values	Default	Description
correlationId	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Correlation ID of message to read.</p> <p>As selection option, you can use correlationId to select specific messages in your queue.</p> <p>MQ defines this field as “unsigned char” that can support binary values. To enter a binary string as the correlationId, use “0x...” as the value. Do not add quote marks around the value.</p>
formatName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>The name of the expected message format. If specified, and the name formatName field of the message does not match, the message is not read. See the requeue option in this table for more information.</p> <p>MQ limits this string to 8 bytes.</p>
groupid	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Group ID of message to read. This is a selection option. MQ defines this field as “unsigned char”, which means that it can support binary values. To enter a binary string as the msgId, use “0x...” as the value. Do not add quote marks around the value.</p>

<i>option values</i>	<i>option_string values</i>	Default	Description
inputMode	<ul style="list-style-type: none"> • browse • Qdefault • shared • exclusive • browse+Qdefault • browse+shared • browse+exclusive 	Qdefault	<p>The values for inputMode open the MQ queue in the following ways:</p> <ul style="list-style-type: none"> • browse – opened for browsing only. The queue manager produces an error when you attempt a destructive read. • Qdefault – opened in the default input mode as defined for the queue. • shared – opened in shared input mode. You receive an error if the queue is already opened in exclusive mode by another MQ handle. • exclusive – opened in exclusive input mode. You receive an error if the queue is already opened in shared or exclusive mode by another MQ handle. • browse+Qdefault – opened for browse- and shared-input mode. • browse+shared – opened for browse- and shared-input mode. You get an error if the queue is already opened in exclusive mode by another MQ handle. • browse+exclusive – opened for browse- and exclusive-input mode. You get an error if the queue is already opened in shared or exclusive mode by another MQ handle. <p>inputMode is valid only for msgrcv.</p> <p>For any endpoint, you must specify inputMode either:</p> <ul style="list-style-type: none"> • On the first msgrcv operation, or • After you specify closeAfterRecv. <p>Attempting to change the value of inputMode across calls may cause unexpected results.</p>
msgld	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Message ID of message to read.</p> <p>As a selection option, you can use msgld to select specific messages in your queue.</p> <p>MQ defines this field as “BYTE array” that can support binary values. To enter a binary string as the msgld, use “0x...” as the value. Do not add quote marks around value, as that is interpreted as a quoted string.</p>
offset	<i>integer</i> between -1, 0 – maxint		<p>Offset of message to read.</p> <p>If -1, the offset is not specified.</p> <p>As selection option, you can use offset to select specific messages in your queue.</p>

option values	option_string values	Default	Description
ordering	<ul style="list-style-type: none"> logical physical 	physical	<p>When ordering is:</p> <ul style="list-style-type: none"> logical – the messages are read in logical order according to groupId, sequenceId, and offsets. physical – the messages are read in the order in which they appear on the queue.
position	<ul style="list-style-type: none"> next cursor 	next	<p>position controls which message is returned. Depending on what inputMode value you specify, there are one or two “read” positions:</p> <ul style="list-style-type: none"> “Normal” – the default read position where destructive reads normally occur. When a queue is opened, the “normal” read position is positioned on the first message in the queue. “Browse cursor” – where the read position has been positioned by a previous call where browse was specified. When a queue is opened for browse, the “browse cursor” is positioned before the first message in the queue. “Browse cursor” is used only for browse+Qdefault, browse+shared, and browse+exclusive <p>If:</p> <ul style="list-style-type: none"> next – the current message at the “normal” read position is returned. The “normal” read position is moved forward to the message after the message returns. cursor – the current message at the “browse cursor” is returned. MQ queue manager raises an error if the “browse cursor” has not yet been positioned. The “browse cursor” is moved forward to the message after the message returns. <p>The MQ queue manager applies the following before determining what message to return:</p> <ul style="list-style-type: none"> The default ordering of the queue (priority or first-in, first-out) Any selection criteria specified (messageId, correlationId, groupId, sequenceId, or offset)

option values	option_string values	Default	Description
requeue	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This must be a full URI of the endpoints.</p> <p>The read message is requeued to the queue specified if:</p> <ul style="list-style-type: none"> • msgrecv reads a message when formatName is specified. • The read message has a different formatName. • requeue is not null. <p>If the message cannot be requeued to the specified queue, the message is left on the queue where it was read, and an exception is raised.</p> <p>MQ limits this string to 48 bytes.</p>
sequenceId	<i>integer</i> between -1, - 9,999,999	-1	<p>Sequence ID of message to read.</p> <p>If -1, the sequence ID is not specified.</p> <p>As a selection option, you can use sequenceId to select specific messages in your queue.</p>
truncationAllowed	<ul style="list-style-type: none"> • yes • no 	no	<p>You can truncate the message when:</p> <ul style="list-style-type: none"> • The buffer used to read the message (bufferLength, or length of the returned datatype). • The buffer is smaller than the length of the message. <p>Specify as:</p> <ul style="list-style-type: none"> • yes – to allow truncation. • no – to not allow truncation. The read fails when the value is no and message is truncated.
timeout	timespec between -1, 0 – (2 ³² -1)	-1	<p>Specifies the timeout.</p> <p>If:</p> <ul style="list-style-type: none"> • -1 – there is no timeout. • timeout is specified as an integer – the value is to be taken in milliseconds. <p>See <i>timespec</i> on page 136 for more information.</p>

JMS – Table 3-9 lists the available *option* and *option_string* values for properties of msgrecv.

Table 3-9: JMS option and option_string values for msgrecv

option values	option_string values	Default	Description
requeue	string	None	The name of a destination, queue, or topic on which to requeue messages that Adaptive Server cannot process. If requeue is not specified, and the message cannot be processed, an error message appears. The endpoint specified must be on the same messaging provider as msgconsume and msgrecv.
timeout	timespec -1, 0 - (2 ³¹ - 1)	-1	By default, msgrecv is a blocking command, which blocks the message until it reads the next message from the message bus. If timeout is not -1, msgrecv returns a null value when the timeout interval lapses without reading a message. The values are in numbers of milliseconds. See timespec on page 136 for more information.

- Unrecognized option names result in an error.

Note This behavior changed with Adaptive Server version 12.5.3a, and differs from earlier versions.

- See @@msgheader on page 30 regarding properties read from the message header.
- msgrecv receives a message from a specified *service_provider* and *service_definition*, and returns that message.
- By default, msgrecv is a blocking command, which blocks the message until it reads the next message from the message bus. If timeout is not -1, msgrecv returns a null value when the timeout interval lapses without reading a message. Its values are in number of milliseconds.
- Adaptive Server handles only messages of types *message*, *text*, or *bytes*. If Adaptive Server encounters a message it cannot process, and requeue is not specified, the message is left on the original queue. Subsequent reads encounter the same message, with the same effect. To prevent this behavior, specify requeue. When you use requeue, messages that Adaptive Server cannot handle are placed on the specified queue.

The specified endpoint must exist on the same messaging service provider as the endpoint used in msgrecv.

- The message includes the binary value of the datatype according to the byte ordering of the host machine.

- Calling `msgrecv` has these results:
 - The value returned is the `message_body` value returned by the message provider, converted to the specified returns type.
 - The values of `@@msgheader` and `@@msgproperties` are set to those of `<msgheader>` and `<msgproperties>` documents, which contain the properties of the message returned by `msgrecv`.
 - You can extract the values of a specific property from a `<msgheader>` and `<msgproperties>` document with `msgpropvalue`. For details, see `msgpropvalue` on page 68.
 - The general format of `<msgheader>` and `<msgproperties>` is described in “Message-related global variables” on page 30.

MQ and `msgrecv`

These are valid only if the provider class is “`ibm_mq`”:

- The `msgId`, `correlationId`, `groupId`, `sequenceId`, and `offset` options act as match criteria for selecting messages. When specified, the next message matching the values specified are returned. The qualification is performed by the WebSphere MQ queue manager.
- If the `MQMD.Format` field of the message received is “MQSTR,” the data is assumed to be character data, and can be returned as `text` or `varchar`. Any other format name can be returned only as `image` or `binary`. One special case is if `MQMD.Format` is “MQHRF.” In this case, the `MQRFH.Format` field is used instead. If the body of the message cannot be returned in the return type specified, the message is sent to the `requeue` option if the `requeue` option is specified; otherwise, the read operation fails. MQ does not enforce that when `MQMD.Format` is “MQSTR,” the message body contains only character data. Programmers should always specify `image` or `varbinary` return types.

Quoting property or option values

- Place apostrophes (') around *option* values to treat them as strings. If you omit the apostrophes, the *option* value is treated as another property name, and the expression is true only if the two properties have the same value.

If your application uses quoted identifiers, the message selector must be enclosed in apostrophes ('). This means that if there are string values in your selectors, you must surround these values with double apostrophes ("). For example:

```
set quoted_identifier on
```

```
select msgrecv ('my_jms_provider?queue=queue.sample',  
  MESSAGE_SELECTOR 'color = 'red''')
```

If your application does not use quoted identifiers, the message selector can be enclosed by ordinary double quotation marks. For example:

```
set quoted_identifier off  
select msgrecv('my_jms_provider?queue=queue.sample',  
  MESSAGE_SELECTOR "color='red'")
```

In this next example, a **messaging client** application sends a message expressing a property named “color” to have the value “red”, and a property named “red” to have the value “color.”

```
select msgsend ('Sending message with property color',  
  'my_jms_provider?queue=queue.sample'  
  MESSAGE_PROPERTY 'color=red, red=color')
```

A client application that wants to consume only messages containing a property named “color” having the value “red” must place double apostrophes (") around the selector value. For example:

```
select msgrecv('my_jms_provider?queue=queue.sample'  
  MESSAGE_SELECTOR 'color='red''')
```

However, the message is not received if the client application uses the following syntax, because “red” is treated as a property name:

```
select msgrecv('my_jms_provider?queue=queue.sample',  
  MESSAGE_SELECTOR 'color=red')
```

In another example, a client sends a message that selects and filters for more than one property:

```
select msgsend('Sending message with properties',  
  'my_jms_provider?queue=queue.sample',  
  MESSAGE_PROPERTY 'color=red, shape=square')
```

If another client wants to select messages in which the property “color” equals “red” and the property “shape” equals “square,” that client must execute the following:

```
select msgrecv('my_jms_provider?queue=queue.sample',  
  MESSAGE_SELECTOR 'color='red'' and shape='square''')
```

Message filters

- If you specify a filter parameter, the filter value is passed directly to the message provider. How it is used depends on the message provider.

- Comparisons specified in the message filter use the sort order specified by the message provider, which may not be the same used by Adaptive Server.
- JMS message providers use a JMS message selector as a filter. The rules for JMS message selectors are:
 - The syntax for the message selector is a subset of conditional expressions, including not, and, or, between, and like.
 - Identifiers are case sensitive.
 - Identifiers must designate message header fields and property names.
- JMS only – if *message_filter* is specified to `msgrecv`, it is ignored.
- MQ only – you can select particular messages by specifying the correlation and the message IDs in the message options.

Permissions

You must have `messaging_role` to run `msgrecv`.

msgsend

Description Provides a SQL interface to send messages to different service endpoints. The endpoints are of type queue.

Syntax

```
message_send_call ::=
    msgsend(message_body, end_point [options_and_properties])
    options_and_properties ::= [option_clause]
    [properties_clause] [header_clause]
    option_clause ::= [,] option_option_string
    properties_clause ::= [,] message_property
    property_option_string
    header_clause ::= [,] message_header
    header_option_string
    message_body ::= scalar_expression |
    (select_for_xml)
    end_point ::= basic_character_expression
```

Parameters

message_body
is the message you are sending. The message body can contain any string of characters. It can be binary data, character data, or SQLX data.

endpoint
is the queue to which a message is addressed. *endpoint* is a *basic_character_expression* where the runtime value is a *service_provider_uri*.

option
allows you to specify options for msgsend. Use the options in Table 3-10 on page 99 if you are using JMS. Use the options in Table 3-11 on page 100 if you are using MQ.

option_string
specifies the general syntax and processing for *option_string*. Individual options are described in the functions that reference them.

```
option_string ::= basic_character_expression
option_string_value ::= option_and_value [ [,] option_and_value ]
option_and_value ::= option_name = option_value
option_name ::= simple_identifier
option_value ::= simple_identifier
    | quoted_string | integer_literal | float_literal | byte_literal
    | true | false | null
```

Parameter	Description
<i>option_string</i>	String describing the option you want to specify
<i>simple_identifier</i>	String that identifies the value of an <i>option</i>
<i>quoted_string</i>	String formed using the normal SQL conventions for embedded quotation marks

Parameter	Description
<i>integer_literal</i>	Literal specified by normal SQL conventions
<i>float_literal</i>	Literal specified by normal SQL conventions
true	A Boolean literal
false	A Boolean literal
null	A null literal
byte_literal	Has the form 0xHH, where each H is a hexadecimal digit

properties_clause

is a *property_option_string*, or one of the options listed in Table 3-12 on page 102 for MQ, and Table 3-13 on page 111 for JMS. The options described in these two tables are set as a property in the message header or message properties, as indicated in the disposition column of the table. The option value is the property value.

Property names are case sensitive.

TIBCO JMS only – if you use a property not listed in Table 3-13 on page 111, it is set as a property in the message properties of the message sent.

Use the options in Table 3-13 on page 111 for msgsend using JMS.

MQ only – the values of *properties_clause* differ based on what you specify in the *rhfCommand* option:

- The properties in Table 3-14 on page 112 are effective only if *rhfCommand* is `deletePublication`.

A `deletePublication` command message sent to the publication stream instructs the MQ pub/sub broker to delete its copy of any retained publications for the specified topics within the publication stream.

The *message_body* argument to msgsend is ignored.

- The properties in Table 3-15 on page 113 are effective only if *rhfCommand* is `deregisterPublisher`.
- The properties in Table 3-16 on page 114 are effective only if *rhfCommand* is `deregisterSubscriber`.

A `deregisterPublisher` command message sent to the MQ pub/sub broker control queue informs the broker that the publisher will no longer publish on the topics specified.

The *message_body* argument to msgsend is ignored.

If the *msgType* is `request`, the reply message is sent to `replyToQmgr` and `replyToQueue`.

- The properties in Table 3-17 on page 115 are effective only if *rhfCommand* is `publish`.

A `publish` command message is sent to the publication stream queue to publish information on specific topics. The publication data is specified as the *message_body* argument to msgsend.

If the `msgType` is `request`, the reply message is sent to `replyToQmgr` and `replyToQueue`.

- The properties in Table 3-19 on page 120 are effective only if `rhfCommand` is `registerSubscriber`.

A `registerSubscriber` command message sent to the MQ pub/sub broker control queue informs the broker that the publisher is publishing, or can, publish data on one or more specified topics. If the publisher is already registered, and there are no other errors, the publisher's registration is modified accordingly.

If the `msgType` is `request`, the reply message is sent to `replyToQmgr` and `replyToQueue`.

- The properties in Table 3-20 on page 123 are effective only if `rhfCommand` is `requestUpdate`.

A `requestUpdate` command message sent to the MQ pub/sub broker control queue informs the broker that the subscriber wants the broker to forward all retained publications that match the topic specified.

If the `msgType` is `request`, the reply message is sent to `replyToQmgr` and `replyToQueue`.

scalar_expression

If a message is a SQL *scalar_expression*, it can be of any datatype.

If the `type` option is not specified, the message type is `text` if the *scalar_expression* evaluates to a character datatype; otherwise, the message type is `bytes`.

If the datatype of the *scalar_expression* is not `character`, it is converted to `varbinary` using the normal SQL rules for implicit conversion. The binary value of the datatype is included in the message according to the byte ordering of the host machine.

basic_character_expression

a Transact-SQL query expression with datatype that is `char`, `varchar`, or `java.lang.String`.

select_for_xml

a `select` expression that specifies a `for xml` clause.

header_clause

allows users to specify only those header properties that are specified in Table 3-12 on page 102 for MQ and Table 3-13 on page 111 for TIBCO JMS. An error displays if you enter an unrecognized header property.

If a recognized header property is specified both in the *message property* and the *message header* clauses, the one in the *message header* clause takes precedence.

An error displays when you specify any unrecognized names in the *message header* parameter.

Examples**Example 1** JMS – sends the message “Hello” to the specified endpoint:

```
select msgsend('Hello', 'my_jms_provider?queue=queue.sample, '
+ 'user=jms_user1,password=jms_user1_password')
```

Example 2 JMS – sends the message “Hello Messaging World!” to the specified endpoint:

```
declare @mymsg varchar (255)
set @mymsg = 'Hello Messaging World!'
select msgsend(@mymsg,
+ 'my_jms_provider?queue=queue.sample,user=jms_user1, '
+ 'password=jms_user1_password')
```

Example 3 TIBCO JMS – sends a message with a body that is a SQLX-formatted representation of the SQL result set, returned by the SQL query to the specified endpoint:

```
select msgsend ((select * from pubs2..publishers FOR XML),
' tibco_jms:tcp://my_jms_host:7222?queue=queue.sample, '
+ 'user=jms_user1,password=jms_user1_password')
```

Example 4 JMS – sets two properties and generates an XML schema for the message:

```
select msgsend
((select pub_name from pubs2..publishers where pub_id = '1389' FOR XML),
my_jms_provider?queue=queue.sample',
MESSAGE PROPERTY 'priority=6, correlationID=MSG_001',
option 'schema=yes')
```

Example 5 JMS – shows user-specified values for message properties:

```
select msgsend ('hello', 'my_jms_provider?queue=queue.sample')
```

```
MESSAGE PROPERTY 'ttl=30,category=5, rate=0.57, rank='top',
priority=6')
```

ttl and priority are internally set as header properties. category, rate, and rank are set as user-specified properties in the message properties.

Example 6 MQ – sends a request message, and the reply is expected on the specified queue, in the same queue manager.

```
select msgsend('do something',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  option 'msgType=request'
MESSAGE PROPERTY 'replyToQueue=QUEUE.REPLY')
```

Example 7 MQ – sends a reply message. The correlation ID, and the reply queue were extracted from a previously received request message:

```
select @correlationId = msgpropvalue("CorrelId", @@msgheader)
select @replyQ = @@msgreplytoinfo
select msgsend('i'm done',
@replyQ
  option 'msgType=report'
  MESSAGE PROPERTY 'correlationId=' + @correlationId)
```

Example 8 MQ – sends a report message. The correlation ID, reply queue, and report message data header were extracted from a previously received request message:

```
select @correlationId = msgpropvalue("CorrelId", @@msgheader)
select @replyQ = @@msgreplytoinfo
select msgsend(@reportData,
@replyQ
  option 'msgType=report'
  MESSAGE PROPERTY 'correlationId=' + @correlationId)
```

Example 9 MQ – sends four datagram messages. Each message is part of the group named “theGroup,” and each message has an increasing sequence number:

```
begin tran
select msgsend('message 1',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'groupId=theGroup,sequenceId=1')
select msgsend('message 2',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
```

```
MESSAGE PROPERTY 'groupId=theGroup,sequenceId=2')
select msgsend('message 3',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'groupId=theGroup,sequenceId=3')
select msgsend('message 4',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'groupId=theGroup,sequenceId=4,lastMsgInGroup=yes')
commit
```

Example 10 MQ – sends a datagram message. Various confirmation reports are requested, and they are sent to the “myReplyQueue:”

```
select msgsend('I want a confirmation',
  'ibm_mq:channel1/TCP/host1(5678)?queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'replyToQueue=myReplyQueue'
  + ',exceptionReport=yes,'
  + ',arrivalReport=withData'
  + ',deliveryReport=withFullData')
```

Example 11 MQ – publishes a datagram message with topics “A”, “A/B”, “A/B/C”. The publisher is registered to publish on topics “A”, “A/B”, and “A/B/C”, and the publication contains information about topic “A/B”. The default MQ pub/sub broker queue and stream queues are used:

```
-- First register the publisher
select msgsend(null,
  'ibm_mq:channel1/TCP/host1(5678)?queue=SYSTEM.BROKER.CONTROL.QUEUE'
  option 'msgType=datagram,rfhCommand=registerPublisher'
  MESSAGE PROPERTY 'topics='a:A/B:a/b/c''')

-- Now publish the publication
select msgsend('something about A/B',
  'ibm_mq:channel1/TCP/host1(5678)?queue=SYSTEM.BROKER.DEFAULT.STREAM'
  option 'msgType=datagram,rfhCommand=publish'
  MESSAGE PROPERTY 'topics=A/B')
```

Example 12 MQ – sends multiple messages in a group. Since ordering is set to logical, specify only the *msgInGroup*, *lastMsgInGroup*, *msgSegment*, *msgLastSegment* options. The queue manager selects a name for the group since it is not specified:

```
begin tran
select msgsend('first logical message of the group',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,msgInGroup=yes')
```



```

select msgsend('second logical message of the group',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,msgInGroup=yes')

select msgsend('third logical message of the group, first segment',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,msgInGroup=yes,msgSegment=yes')

select msgsend('third logical message of the group, second segment',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,msgInGroup=yes,msgSegment=yes')

select msgsend('third logical message of the group, third segment',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,msgInGroup=yes,msgLastSegment=yes')

select msgsend('fourth logical message of the group',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
  MESSAGE PROPERTY 'ordering=logical,lastMsgInGroup=yes')
commit

```

Example 13 Uses the `alter_user=yes` option in `msgsend` to allow user Joe—whose SQL login is “joe”—to send and receive messages to and from the MQ application running on machine “host1” through Adaptive Server, even though there is no user ID called “joe” on host1.

```

select msgsend('Hello world',
  'ibm_mq:chnl1/TCP/host1(5678)?qmgr=joeQM,queue=QUEUE1,alter_user=yes')

```

Example 14 Uses `msgsend` to register, then deregister a subscriber. The subscriber is interested in all publications that match the topics “A” or “A/B/*”. Matching publications are forwarded to the queue “Q2” by the MQ pub/sub broker:

```

-- Register the subscriber
select msgsend(null,
  'ibm_mq:channel1/TCP/host1(5678)'
  + '?qmgr=QM,queue=SYSTEM.BROKER.CONTROL.QUEUE'
  option 'msgType=datagram,rfhCommand=registerSubscriber'
  MESSAGE PROPERTY 'topics='A:A/B/*',streamName=stream1,queueName=Q2')

-- Publish a message to the stream queue, let it do implicit registration
select msgsend('happy birthday',
  'ibm_mq:channel1/TCP/host1(5678)?qmgr=QM,

```

```
        queue=stream1'
option 'msgType=datagram,rfhCommand=publish'
MESSAGE PROPERTY 'topics='A''')

-- Read a message forwarded to us by the MQ pub/sub
select msgrecv(
  'ibm_mq:channel1/TCP/host1(5678)?qmgr=QM,
    queue=Q2'
option 'timeout=50ss')

-- Deregister the subscriber
select msgsend(null,
  'ibm_mq:channel1/TCP/host1(5678)'
    + ?qmgr=QM,queue=SYSTEM.BROKER.CONTROL.QUEUE'
option 'msgType=datagram,rfhCommand=deregisterSubscriber'
MESSAGE PROPERTY 'topics='A:A/B/*'',streamName=stream1,queueName=Q2')
```

Usage

- If the destination has the form `queue=queue_name`, the message is sent to this queue.
- The `service_provider_class` and the words “user” and “password” are case insensitive. `local_name`, `hostname`, `port`, `queue_name`, `user_name`, and `password` parameters are case sensitive.
- You can set message properties specific to Adaptive Server according to Table 3-3 on page 38.
- Option string usage in `msgsend`:
 - Empty option strings are ignored.
 - You can separate option strings with commas or white space (there is no limit on the amount of white space before first option, after the last option, between options, and surrounding the equal signs).
 - Quoted strings are formed according to SQL conventions for embedded quotation marks.
 - If you specify multiple options with the same name, only the option listed last is processed. For example, in the following statement, only the value 7 is used or validated for 'priority'; other values are ignored:

```
select msgsend( 'Hello Messaging World!',
  'my_jms_provider?queue=queue.sample',
  MESSAGE PROPERTY 'priority='high'', priority=yes, priority=7')
```

- After you execute `msgsend`, the values of the global variables are set with information for that call. For more details, see “Message-related global variables” on page 30.
- Use single apostrophes ('), not double quotation marks ("), around quoted option or property values.

Note `msgsend` also allows messages to be sent to a topic, if you specify `topic=topic_name` as the destination. Sybase does not recommend this practice, as it may cause unexpected behavior.

- Unrecognized options or properties are ignored, but unrecognized option or property values are flagged as an error.

Note This behavior changed with Adaptive Server version 12.5.3a, and differs from earlier versions.

`msgsend` option *option_string* parameter values

Table 3-10 lists the available `msgsend` option parameters for JMS.

Table 3-10: Valid JMS option *option_string* types and values for *msgsend*

Types	Values	Default	Description
schema	<ul style="list-style-type: none"> • no • yes • “<i>user_schema</i>” 	no	<ul style="list-style-type: none"> • <i>user_schema</i> is a user-supplied schema describing the <i>message_body</i>. • no indicates that no schema is generated and sent out as part of the message. • yes indicates that Adaptive Server generates an XML schema for the message. <i>yes</i> is meaningful only in a <i>message_body</i> that uses the parameter <i>select_for_xml</i>. <i>select_for_xml</i> generates a SQLX-formatted representation of the SQL result set. The generated XML schema is a SQLX-formatted schema that describes the result set document. <p>The schema is included in the message as the ASE_MSGBODY_SCHEMA property.</p>
type	text, bytes	text	The type of message to send.

Table 3-11 lists the available `msgsend` option parameters for MQ.

Table 3-11: Valid MQ option `option_string` types and values for `msgsend`

Types	Values	Default	Description
msgType	<ul style="list-style-type: none">• datagram• request• reply• report	datagram	If the type of the message is: <ul style="list-style-type: none">• request – you must also specify the <code>replyQueue</code> property.• report – you must also specify the <code>reportDataHeader</code> and <code>feedback</code> properties.

Types	Values	Default	Description
rfhCommand	<ul style="list-style-type: none"> • null • deletePublication • deregisterPublisher • deregisterSubscriber • publish • registerPublisher • registerSubscriber • requestUpdate 	null	<p>MQRF headers, for MQ pub/sub, are control messages that are sent to a queue and read by the MQ pub/sub broker. The broker acts upon the message it reads from the queue.</p> <p>If rfhCommand is null, the message does not include the MQRF header. The message includes the MQRF header with any other value for rfhCommand, with the MQPSCCommand set to the following:</p> <ul style="list-style-type: none"> • deletePublication – set to DeletePub. The endpoint is the endpoint to the publishing stream queue. See Table 3-14 on page 112. • deregisterPublisher – set to DeregPub. See Table 3-15 on page 113. • deregisterSubscriber – set to DeleteSub. See Table 3-16 on page 114. • publish – set to Publish. The endpoint is the endpoint to the publishing stream queue. See Table 3-17 on page 115. • registerPublisher – set to RegPub. See “msgsend properties if rfhCommand is set to deletePublications” on page 112. • registerSubscriber – set to RegSub. See Table 3-19 on page 120. • requestUpdate – set to ReqUpdate. See Table 3-20 on page 123. <p>The message is sent to the endpoint you specify. For these options, specify the endpoint to the publishing stream queue:</p> <ul style="list-style-type: none"> • publish • deletePublication <p>For these options, specify the endpoint to the MQ pub/sub broker control queue:</p> <ul style="list-style-type: none"> • deregisterPublisher • deregisterSubscriber • registerPublisher • registerSubscriber • requestUpdate

Types	Values	Default	Description
alter_user	<ul style="list-style-type: none"> • yes • no 	null	<p>The alter_user=yes option allows users who were granted messaging_role permission to send and receive messages from a machine running MQ, even if they do not have an operating system (login) ID on that machine.</p> <p>If you do not set this option and the user does not have a login ID on the machine running MQ, the MQ authentication fails and the messaging operation does not succeed.</p> <hr/> <p>Note If the machine running MQ is not also running Adaptive Server Enterprise, users see an error message even after running alter_user=yes. To prevent this, create a new login on the MQ machine that is identical to the user ID of the user that started Adaptive Server.</p>

msgsend *properties_clause* parameter values

Table 3-12 lists the available msgsend *properties_clause* parameters for MQ.

Table 3-12: Valid MQ message property property_option_clause types and values for msgsend

Types	Values	Default	Description
arrivalReport	<ul style="list-style-type: none"> • yes • withData • withFullData • no 	no	<p>Arrival of this message to the final destination should generate a confirm-on-arrival (COA) report. You must specify replyToQueue. If you specify:</p> <ul style="list-style-type: none"> • yes – the COA report generates without data from the received message. • withData – the COA report generates with the first 100 bytes of the data from the received message. • withFullData – the COA report generates with the full data from the received message. • no – the COA report is not generated.

Types	Values	Default	Description
correlationId	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Clients set correlation ID to link messages together. MQ limits this string to 24 bytes.</p> <p>MQ defines this field as unsigned char, which indicates that it can support binary values. To enter a binary string as the correlationId, use “0x...” as the value.</p> <p>Do not use quotes around the value.</p> <p>If <i>rfhCommand</i> is not null:</p> <ul style="list-style-type: none"> • If <i>correlationId</i> is not null, a new correlation ID is not requested. If <i>correlationAsId</i> is yes, and <i>correlationId</i> is null, this is a separate traditional identity (one where correlation ID is empty). • For <i>rfhCommands</i> of <i>deletePublication</i>, <i>deregisterPublisher</i>, <i>publish</i>, and <i>registerPublisher</i>, the correlation ID specified is as part of the publisher’s traditional identity.
deliveryReport	<ul style="list-style-type: none"> • yes • <i>withData</i> • <i>withFullData</i> • no 	no	<p>Delivery of this message from the final destination generates a confirm-on-delivery (COD) report.</p> <p>You must specify <i>replyToQueue</i>. If:</p> <ul style="list-style-type: none"> • <i>yes</i> – the COA report generates without data from the received message. • <i>withData</i> – the COA report generates with the first 100 bytes of the data from the received message. • <i>withFullData</i> – the COA report generates with the full data from the received message. • <i>no</i> – the COA report is not generated.
exceptionReport	<ul style="list-style-type: none"> • yes • <i>withData</i> • <i>withFullData</i> • no 	no	<p>Expiration of this message or failure of this send generates an exception report.</p> <p>You must specify <i>replyToQueue</i>. If:</p> <ul style="list-style-type: none"> • <i>yes</i> – the exception report generates without data from the received message. • <i>withData</i> – the exception report generates with the first 100 bytes of the data from the received message. • <i>withFullData</i> – the exception report generates with the full data from the received message. • <i>no</i> – the exception report is not generated.

Types	Values	Default	Description
expirationReport	<ul style="list-style-type: none"> • yes • withData • withFullData • no 	no	<p>The failure of this send generates an exception report.</p> <p>You must specify replyToQueue. If:</p> <ul style="list-style-type: none"> • yes – the exception report generates without data from the received message. • withData – the exception report generates with the first 100 bytes of the data from the received message. • withFullData – the exception report generates with the full data from the received message. • no – the exception report is not generated.
expiry	timespec between -1 and 214748364799	-1, no expiration	<p>The message's time-to-live on the queue manager. Units are in milliseconds if the timespec is an integer.</p> <p>Values are:</p> <ul style="list-style-type: none"> • 0 – message does not expire. • -1 – uses the default defined for the queue. <hr/> <p>Note expiry is in tenths of a second, so this number is rounded to the tenths of a second before being passed to MQ.</p> <hr/> <p>See timespec on page 136 for more information.</p>
feedback	<p><i>integer</i></p> <p>Must range within MQFB_APPL_FIRST (65536) to MQFB_APPL_LAST (99999999)</p>	0	<p>For report messages, feedback is a code that indicates the nature of the report message.</p> <p>MQ defines one feedback code range each for:</p> <ul style="list-style-type: none"> • System report messages • Application report messages
formatName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Application-defined property to pass information about the message formats.</p> <p>This property allows sending applications to set a format name that describes the message data.</p> <p>A receiving application can check formatName in <i>@msgheader</i> to decide how to process the message data.</p> <p>Names beginning with "MQ" are reserved.</p> <p>MQ limits this string to 8 bytes.</p>

Types	Values	Default	Description
groupID	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>User-defined group.</p> <p>MQ limits this string to 24 bytes.</p> <p>MQ defines this field as unsigned char, which indicates that it can support binary values. To enter a binary string as the groupID, use “0x...” as the value. Do not use quotes around the value, or it is interpreted as a quoted string.</p> <p>If groupID is not specified and one of the grouping properties is specified, the queue manager generates the group name.</p> <p>Ignored if ordering is set to logical.</p> <p>All messages of a group must be sent in the same transaction.</p>
lastMsgInGroup	<ul style="list-style-type: none"> • yes • no 	no	<p>If the value is yes, marks a message as being the last logical message of a group.</p> <p>To have a single logical message in a group by itself, you must set lastMsgInGroup to yes.</p> <p>You must send all messages of a group in the same transaction.</p>
mode	<ul style="list-style-type: none"> • persistent • non-persistent • default 	default	<p>If mode is:</p> <ul style="list-style-type: none"> • persistent – the message is backed by the messaging provider, using stable storage. If the messaging provider crashes before the message can be consumed, the message is lost, unless mode is set to persistent. • non-persistent and the messaging provider crashes – you may lose a message before it reaches the desired destination. • default – the default defined for the queue is used.
msgID	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>When specified, WebSphere MQ replaces any existing message ID with the value specified for msgID.</p> <p>MQ limits this string to 24 bytes.</p> <p>MQ defines this field as “unsigned char,” which indicates that it can support binary values.</p> <p>To enter a binary string as the msgID, use “0x...” as the value. Do not use quotes around the value.</p>

Types	Values	Default	Description
msgInGroup	<ul style="list-style-type: none"> • yes • no 	no	<p>If the value is yes, this message is a logical message of a message group.</p> <p>For messages in a group, you must set this property to yes for all logical messages of the group, except the last one, which should have lastMsgInGroup set to yes.</p> <p>You must send all messages of a group in the same transaction.</p>
msgLastSegment	<ul style="list-style-type: none"> • yes • no 	no	<p>If the value is yes, this message is the last segment of a segmented message. To have a segment message in a local message by itself, the message must have msgLastSegment set to yes.</p> <p>When the value is yes and ordering is set to physical, you must also set the offset property.</p> <p>You must send all messages in a group in the same transaction.</p>
msgSegment	<ul style="list-style-type: none"> • yes • no 	no	<p>If the value is yes, this message is a segment of a segmented message. For messages that are part of a single segment, you must set this property to yes for all segments except the last one, which should be have msgLastSegment set to yes.</p> <p>When the value is yes and ordering is set to physical, you must also set the offset property.</p> <p>You must send all messages in a group in the same transaction.</p>
negativeActionReport	<ul style="list-style-type: none"> • yes • no 	no	<p>You must specify replyToQueue. If:</p> <ul style="list-style-type: none"> • yes – when the retrieving application reads this message and acts negatively on it, a negative-action (NAN) report is generated. • no – the NAN report is not generated.
offset	<i>integer</i> between -1, 0 – maxint	-1	<p>When the message is a segment of a segmented message, set offset to the byte offset of the current message within the logical message.</p> <p>-1 indicates that the offset is not specified.</p> <p>offset is ignored unless msgSegment, or msgLastSegment are also specified.</p> <p>Ignored by msgpublish.</p> <p>Ignored if ordering is set to logical.</p> <p>You must send all messages of a group in the same transaction.</p>

Types	Values	Default	Description
onNoDelivery	<ul style="list-style-type: none"> deadLetter discard 	deadLetter	<p>If:</p> <ul style="list-style-type: none"> deadLetter – if the message cannot be delivered, the message is put on the dead-letter queue. discard – the message is discarded by the queue manager.
ordering	<ul style="list-style-type: none"> logical physical 	physical	<p>When this property is:</p> <ul style="list-style-type: none"> physical – the application can send messages that are part of a group (or segmented message) in any order. The queue manager returns errors if it detects missing segments, or holes in the sequence identifiers. logical – the application needs only to set the msgInGroup, lastMsgInGroup, msgSegment, and lastMsgSegment options appropriately. The queue manager automatically sets the group name, sequence identifier, and segment offset.
positiveActionReport	<ul style="list-style-type: none"> yes no 	no	<p>You must specify replyToQueue. If:</p> <ul style="list-style-type: none"> yes – when the retrieving application reads this message and acts positively on it, a positive-action notification (PAN) report is generated. no – the PAN report is not generated.
priority	<p><i>integer:</i></p> <ul style="list-style-type: none"> -1, 0 to queue manager configured max priority 	-1	<p>Controls the priority of the message. If:</p> <ul style="list-style-type: none"> -1 – the default priority as defined for the queue is used. priority specified is greater than the max priority defined for the queue manager – the max priority defined for the queue manager is used. This is implemented by MQ.
replyCorrelationId	<ul style="list-style-type: none"> msgId correlationId 	msgId	<p>If:</p> <ul style="list-style-type: none"> msgId – the correlation ID in the report message uses the message ID of the received message. correlationId – the correlation ID in the report message uses the correlation ID of the received message.
replyMsgId	<ul style="list-style-type: none"> new original 	new	<p>If:</p> <ul style="list-style-type: none"> new – the generated report message contains a new message ID. original – the report message uses the same message ID as the message received.

Types	Values	Default	Description
replyToInputMode	<ul style="list-style-type: none"> • browse • Qdefault • shared • exclusive • browse+Qdefault • browse+shared • browse+exclusive 	Qdefault	<p>The mode that the replyToQueue is opening.</p> <p>When you specify replyToQueue, the queue is automatically opened for subsequent input. This mode specifies the input mode that the replyToQueue is opening.</p> <p>This property is ignored if you do not specify replyToQueue.</p> <p>The modes have the following meanings:</p> <ul style="list-style-type: none"> • browse – the queue is opened for browsing only. An error displays from the queue manager if you attempt to perform a destructive read. • Qdefault – the queue is opened in the default input mode as defined for the queue. • shared – the queue is opened in shared input mode. An error displays if the queue is already opened in exclusive mode by another MQ handle. • exclusive – the queue is opened in exclusive input mode. An error displays if the queue is already opened in shared or exclusive mode by another MQ handle. • browse+Qdefault – the queue is opened for browsing, as well as for the default input mode as defined for the queue. • browse+shared – the queue is opened for browsing, as well as for shared input mode. An error displays if the queue is already opened in exclusive mode by another MQ handle. • browse+exclusive – the queue is opened for browsing, as well as for exclusive input mode. An error displays if the queue is already opened in shared or exclusive mode by another MQ handle.
replyToModel	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>The name of the model queue from which the reply queue is created, when the replyToQueue is a dynamic queue.</p> <p>If you do not specify replyToQueue, this property is ignored.</p> <p>MQ limits this string to 48 bytes.</p>

Types	Values	Default	Description
replyToQmgr	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>The queue manager where replyToQueue resides.</p> <p>If you do not specify replyToQueue, this property is ignored.</p> <p>MQ limits this string to 48 bytes.</p>
replyToQueue	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>The queue where the application expects a reply to a request message.</p> <hr/> <p>Note The message type sent does not have to be request, as MQ does not enforce this.</p> <hr/> <p>If replyToQmgr is not specified, replyToQueue is assumed to be the same queue manager as the current endpoint.</p> <p>If the queue name specified ends with a “*”, a system-generated dynamic queue name is generated with the specified prefix.</p> <p>If replyToModel and a dynamic queue name are specified, the dynamic queue is created from the model queue specified for replyToModel.</p> <p>You can obtain system-generated dynamic queue names after the send operation via the <code>@msgreplytoinfo</code> session variable.</p> <hr/> <p>Note When you specify a dynamic queue name, the current Adaptive Server login must have “crt” authorization in the queue manager to create the dynamic queue.</p> <hr/> <p>When a dynamic queue name is specified, you must manually delete the dynamic queue that is created if the receiving application does not do so.</p> <p>When rfhCommand is not null, you can specify replyToQueue to get responses from the MQ pub/sub broker.</p>

Types	Values	Default	Description
rfhCommand	<ul style="list-style-type: none"> • null • deletePublication • deregisterPublisher • deregisterSubscriber • publish • registerPublisher • registerSubscriber • requestUpdate 	null	<p>MQRF headers, for MQ pub/sub, are control messages that are sent to a queue and read by the MQ pub/sub broker. The broker acts upon the message that it reads from the queue.</p> <p>If rfhCommand is null, the message does not include the MQRF header. The message includes the MQRF header with any other value for rfhCommand, with the MQPSCCommand set to the following:</p> <ul style="list-style-type: none"> • deletePublication – set to DeletePub. The endpoint is the endpoint to the publishing stream queue. See Table 3-14 on page 112. • deregisterPublisher – set to DeregPub. See Table 3-15 on page 113. • deregisterSubscriber – set to DeleteSub. See Table 3-16 on page 114. • publish – set to Publish. The endpoint is the endpoint to the publishing stream queue. See Table 3-17 on page 115. • registerPublisher – set to RegPub. See “msgsend properties if rfhCommand is set to deletePublications” on page 112. • registerSubscriber – set to RegSub. See “msgsend properties if rfhCommand is set to deletePublications” on page 112. • requestUpdate – set to ReqUpdate. See “msgsend properties if rfhCommand is set to deletePublications” on page 112. <p>The message is sent to the endpoint you specify. For these options, specify the endpoint to the publishing stream queue:</p> <ul style="list-style-type: none"> • publish • deletePublication <p>For these options, specify the endpoint to the MQ pub/sub broker control queue:</p> <ul style="list-style-type: none"> • deregisterPublisher • deregisterSubscriber • registerPublisher • registerSubscriber • requestUpdate

Types	Values	Default	Description
sequenceId	integer between -1 – 9,999,999	-1	Used to sequence logical messages that are part of a group. -1 indicates that the sequenceId is not specified. sequenceId is ignored unless msgInGroup or lastMsgInGroup are also specified. Ignored by msgpublish. Ignored if ordering is set to logical. You must send all messages of a group in the same transaction.

Table 3-13 lists the available msgsend *properties_clause* parameters for JMS.

Table 3-13: Valid JMS message property *properties_option_string* types and values for msgsend

Option	Values	Default	Disposition	Description
ttl	0 - (2 ⁶³ - 1)	0	header	ttl refers to time-to-live on the messaging bus. Adaptive Server is not affected by this. Expiry information is the duration of time in milliseconds during which a message is valid. For instance, 60 indicates that the life of the message is 60 milliseconds. A value of 0 indicates that the message never expires. ttl uses the timespec option. See timespec on page 136 for more information.
priority	1 to 9	4	header	The behavior of priority is controlled by the underlying message bus. The values mentioned here apply to TIBCO JMS. Priorities from 1 to 4 are normal; priorities from 5 to 9 are expedited.
correlation	string	none	header	Client applications set correlation IDs to link messages together. Adaptive Server sets the correlation ID the application specifies.

Option	Values	Default	Disposition	Description
mode	<ul style="list-style-type: none"> persistent non-persistent 	persistent	header	<p>If the mode is:</p> <ul style="list-style-type: none"> persistent – the message is backed by the JMS provider, using stable storage. If the messaging provider crashes before the message is consumed, the message is lost, unless mode is set to persistent. non-persistent and the messaging provider crashes – you may lose a message before it reaches the desired destination.
replyqueue	A string containing a <i>queue_name</i>	none	header	<p>If the value of <i>queue_name</i> or <i>topic_name</i> is:</p> <ul style="list-style-type: none"> syb_temp – Adaptive Server creates a temporary destination and sends information related to the newly created temporary destination as a part of the header information. The type of the temporary destination, queue or topic, depends on whether you specify replyqueue or replytopic. Only the option listed last is used. A destination that already exists – Adaptive Server does not create a new destination, using instead the one specified by the user.
replytopic	A string containing a <i>topic_name</i>	none	header	

msgsend properties and *rfhCommand*

For MQ, properties in Table 3-14 are effective only if *rfhCommand* is *deletePublication*.

Table 3-14: *msgsend* properties if *rfhCommand* is set to *deletePublications*

Property	Values	Default	Description
local	<ul style="list-style-type: none"> yes no 	no	<p>If:</p> <ul style="list-style-type: none"> yes – only the retained publications published locally at this broker are deleted. no – globally retained publications are deleted from all brokers in the network.
streamName	<ul style="list-style-type: none"> null <i>string</i> 	null	<p>Name of the publication stream for the specified topics.</p> <p>If not specified, the default is the stream queue to which this MQRFH command message is sent. MQ limits this string to 48 bytes.</p>

Property	Values	Default	Description
topics	<i>string</i>	none	Use the format detailed in “Syntax for topics” on page 16. Retained messages matching this topic are deleted. At least one topic must be supplied. This is a required property, and is an error if omitted.

For MQ, properties in Table 3-14 are effective only if `rhfCommand` is `deregisterPublisher`.

Table 3-15: *msgsend* properties if *rhfCommand* is set to *deregisterPublisher*

Property	Values	Default	Description
deregAll	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – all topics registered for this publisher are deregistered, and the topics property is ignored. • no – no registered topics are deregistered. <p>Adaptive Server returns an error if you specify topics.</p>
streamName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>If:</p> <ul style="list-style-type: none"> • Not null – this is the name of the publication stream. • null – SYSTEM.BROKER.DEFAULT.STREAM is assumed. <p>MQ limits this string to 48 bytes.</p>
topics	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Use the format detailed in “Syntax for topics” on page 16.</p> <p>These are the topics that this publisher deregisters.</p> <p>Adaptive Server returns an error if:</p> <ul style="list-style-type: none"> • The <code>deregAll</code> property is set to yes. • <code>topics</code> is not null.
qmgrName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the publisher’s queue manager name, used to establish the publisher’s traditional identity. Specify it as the same value you specified when you registered the publisher.</p> <p>If null, defaults to <code>replyToQmgr</code>.</p>

Property	Values	Default	Description
queueName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the publisher's queue name, used to establish the traditional identity of the publisher. Specify it as the same value you specified when you registered the publisher.</p> <p>If null, defaults to the replyToQueue.</p>
correlationAsId	<ul style="list-style-type: none"> • yes • no • generate 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – correlationId is used as part of the publisher's traditional identity. You must specify correlationId, but not as 0x00. • no – correlationId is not used as part of the publisher's traditional identity. • generate – a system-generated correlationId is used as part of the publisher's traditional identity.

For MQ, the properties in Table 3-16 are effective only if `rhfCommand` is `deregisterSubscriber`.

Table 3-16: msgsend properties if `rhfCommand` is set to `deregisterSubscriber`

Property	Values	Default	Description
deregAll	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – all topics for this subscriber are deregistered. The topics property is ignored. • no – no subscriber topics are deregistered. <p>Adaptive Server returns an error if topics are not null</p>
streamName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>If:</p> <ul style="list-style-type: none"> • Not null – this is the name of the publication stream. • null – <code>SYSTEM.BROKER.DEFAULT.STREAM</code> is assumed. <p>MQ limits this string to 48 bytes.</p>

Property	Values	Default	Description
topics	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>Use the format detailed in “Syntax for topics” on page 16.</p> <p>These are the topics that this subscriber deregisters.</p> <p>Adaptive Server returns an error if:</p> <ul style="list-style-type: none"> • deregAll is Yes. • topics are not null.
qmgrName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the subscriber’s queue manager name, used to establish the traditional identity of the subscriber. Specify it as the same value that was specified when you registered the subscriber.</p> <p>If null, it defaults to the replyToQmgr.</p>
queueName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the subscriber’s queue name, used to establish the traditional identity of the subscriber. Specify it as the same value that was specified when you registered the subscriber.</p> <p>If null, it defaults to the replyToQueue.</p>
correlationAsId	<ul style="list-style-type: none"> • yes • no • generate 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – correlationId is used as part of the publisher’s traditional identity. You must specify correlationId, but not as 0x00. • no – correlationId is not used as part of the publisher’s traditional identity. • generate – a system-generated correlationId is used as part of the publisher’s traditional identity.

For MQ, the properties in Table 3-17 are effective only if `rhfCommand` is `publish`.

Table 3-17: msgsend properties if `rhfCommand` is set to `publish`

Property	Values	Default	Description
topics	<i>string</i>	none	<ul style="list-style-type: none"> • Use the format detailed in “Syntax for topics” on page 16. • Wildcards are not allowed. • These are the topics on which this publication has information. • This is a required property, and generates an error if omitted.

Property	Values	Default	Description
anon	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the identity of the publisher is not divulged by the MQ pub/sub broker. Ignored if noReg is yes. • no – the identity of the publisher is divulged by the MQ pub/sub broker.
local	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the MQ pub/sub broker sends this publication only to subscribers that registered specifying local. Ignored if noReg is yes. • no – the MQ pub/sub broker sends this publication to all subscribers.
directReq	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the publisher is willing to accept direct request for publication information from other applications. Ignored if noReg is yes. <p>Do not set this option to yes if the anon property is also set to yes, since the MQ pub/sub broker responds with an error.</p> <ul style="list-style-type: none"> • no – the publisher is not willing to accept direct request for publication information from other applications.
noReg	<ul style="list-style-type: none"> • yes • no 	no	<p>If the publisher is not already registered with the MQ pub/sub broker as a publisher for this stream and topic and the value of NoReg is:</p> <ul style="list-style-type: none"> • yes – the MQ pub/sub broker does not perform an implicit registration. The anon, local, and directReq properties are ignored. • no – the MQ pub/sub broker performs an implicit registration, using the values set by anon, local, and directReq. <p>If the publisher is already registered, and anon, local, or directReq are set to yes, the existing registration is altered according to those properties.</p>

Property	Values	Default	Description
otherSubsOnly	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the MQ pub/sub broker sends this publication to this publisher if this publisher has a subscription on this publication. • no – the MQ pub/sub broker does not send this publication to this publisher, even if this publisher has a subscription on this publication.
publishSequenceId	<i>number</i> between -1, 0–($2^{32} - 1$)	-1	<p>If:</p> <ul style="list-style-type: none"> • Not -1, this is the sequence number of the publication. It should increase with each publication, but the MQ pub/sub broker does not validate it. • If -1, the sequence number is not set.
publishTimeStamp	<ul style="list-style-type: none"> • null • <i>integer</i> 	null	<p>If:</p> <ul style="list-style-type: none"> • Not null, this is the publication timestamp in the form of YYYYMMDDHHMMSSth, using universal time. The format is not validated. • null – the publication timestamp is not set.
qmgrName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the queue manager used to determine the publisher’s traditional identity. This is also where subscribers can send direct requests to this publisher.</p> <p>MQ limits this string to 48 bytes.</p>
queueName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the queue used to determine the publisher’s traditional identity. This is also where subscribers can send direct requests to this publisher.</p> <p>MQ limits this string to 48 bytes.</p>
retainPub	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the MQ pub/sub broker does not send this publication to this publisher, even if this publisher has a subscription on this publication. • no – the MQ pub/sub broker sends this publication to this publisher if this publisher has a subscription on this publication.

Property	Values	Default	Description
stringData	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>If not null, this is optional publisher-defined information that is included in the publication's MQRF header.</p> <hr/> <p>Note Although MQ pub/sub allows multiple stringData tags in the MQRF header, RTDS supports only one.</p>
integerData	<i>number</i> between -1, 0–($2^{32} - 1$)	-1	<p>If not -1, this is optional publisher-defined information that is included in the publication's MQRF header.</p> <hr/> <p>Note Although MQ pub/sub allows multiple integerData tags in the MQRF header, RTDS supports only one.</p>
correlationAsId	<ul style="list-style-type: none"> • yes • no • generate 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – correlationId is used as part of the publisher's traditional identity. You must specify correlationId, but not as 0x00. • no – correlationId is not used as part of the publisher's traditional identity. • generate – a system-generated correlationId is used as part of the publisher's traditional identity.

For MQ, the properties in Table 3-18 are effective only if `rhfCommand` is `registerPublisher`.

Table 3-18: MQ msgsend properties if `rhfCommand` is set to `registerPublisher`

Property	Values	Default	Description
anon	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – MQ pub/sub broker does not divulge the identity of the publisher. • no – MQ pub/sub broker divulges the identity of the publisher.

Property	Values	Default	Description
correlationAsId	<ul style="list-style-type: none"> • yes • no • generate 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – correlationId is used as part of the publisher’s traditional identity. You must specify correlationId, but not as 0x00. • no – correlationId is not used as part of the publisher’s traditional identity. • generate – a system-generated correlationId is used as part of the publisher’s traditional identity.
directReq	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the publisher is willing to accept direct request for publication information from other applications. <p>Do not set this option to yes if the anon property is also set to yes, since the MQ pub/sub broker responds with an error.</p> <ul style="list-style-type: none"> • no – the publisher is not willing to accept direct request for publication information from other applications.
local	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the MQ pub/sub broker sends this publication only to subscribers that registered specifying Local. • no – the MQ pub/sub broker sends this publication to all subscribers.
qmgrName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the queue manager used to determine the publisher’s traditional identity. This is also where subscribers can send direct requests to this publisher.</p> <p>MQ limits this string to 48 bytes.</p>
queueName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>This is the queue used to determine the publisher’s traditional identity. This is also where subscribers can send direct requests to this publisher.</p> <p>MQ limits this string to 48 bytes.</p>

Property	Values	Default	Description
streamName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	<p>If:</p> <ul style="list-style-type: none"> • Not null – this is the stream where the publisher publishes publications. • null – the default is SYSTEM.BROKER.DEFAULT.STREAM. <p>MQ limits this string to 48 bytes.</p>
topics	<i>string</i>	none	<p>Use the format detailed in “Syntax for topics” on page 16.</p> <p>Wildcards are not allowed.</p> <p>These are the topics on which the publisher provides information on.</p> <p>This is a required property, and generates an error if omitted.</p>

For MQ, the properties in Table 3-19 are effective only if `rhfCommand` is `registerSubscriber`.

Table 3-19: MQ msgsend properties if `rhfCommand` is set to `registerSubscriber`

Property	Values	Default	Description
topics	<i>string</i>	none	<p>Use the format detailed in “Syntax for topics” on page 16.</p> <p>These are the topics on which the subscriber wants to receive publications.</p> <p>This is a required property, and generates an error if omitted.</p>
anon	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – MQ pub/sub broker does not divulge the identity of the subscriber. • no – MQ pub/sub broker divulges the identity of the subscriber.
local	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the subscription is not distributed to other brokers in the network. Only publications published from this node by a publisher specifying Local are sent to this subscriber. • no – the subscription is not specified in the RFH command.

Property	Values	Default	Description
newPubsOnly	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the broker sends this publication only to this subscriber, and retained publications that exist at registration time are not sent. • no – the publication is not specified in the RFH command.
pubOnReqOnly	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the broker sends only new publications to this subscriber. Retained publications that exist at registration time are not sent. • no – the publication is not specified in the RFH command.
inclStreamName	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the broker adds the publication stream name in the MQRF header to each message that is forwarded to the subscriber. • no – the publication is not specified in the RFH command.
informIfRet	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the broker informs the subscriber if the publication is retained, by setting the MQPSPubsOptslsRetainedPub in the MQRF header of the message sent to the subscriber. • no – the publication is not specified in the RFH command.
dupsOk	<ul style="list-style-type: none"> • yes • no 	no	<p>If:</p> <ul style="list-style-type: none"> • yes – the broker is allowed to occasionally deliver a duplicate publication to the subscriber. • no – the publication is not specified in the RFH command.

Property	Values	Default	Description
pubsPersistence	<ul style="list-style-type: none"> non-persistent persistent asPublication asQueue 	asQueue	<p>If:</p> <ul style="list-style-type: none"> non-persistent – the publication is placed on the subscriber queue as a non-persistent message. persistent – the publication is placed on the subscriber queue as a persistent message. asPublication – the publication is placed on the subscriber queue with the same persistence as the original publication. asQueue – the publication is placed on the subscriber queue with the default persistence of the subscriber queue.
streamName	<ul style="list-style-type: none"> null string 	null	<p>If:</p> <ul style="list-style-type: none"> Not null – this is the stream where the publisher publishes publications. null – the subscription is identified by its traditional identity.
qmgrName	<ul style="list-style-type: none"> null string 	null	<p>This is the queue manager used to determine the subscriber's traditional identity. MQ limits this string to 48 bytes.</p>
queueName	<ul style="list-style-type: none"> null string 	null	<p>This is the queue used to determine the subscriber's traditional identity. MQ limits this string to 48 bytes.</p>
correlationAsId	<ul style="list-style-type: none"> yes no generate 	no	<p>If:</p> <ul style="list-style-type: none"> yes – correlationId is used as part of the subscriber's traditional identity. You must specify correlationId, but not as 0x00. no – correlationId is not used as part of the subscriber's traditional identity. generate – a system-generated correlationId is used as part of the subscriber's traditional identity.

The properties in Table 3-20 are effective only if rhfCommand is requestUpdate.

Table 3-20: MQ msgsend properties if rfhCommand is set to requestUpdate

Property	Values	Default	Description
topics	<i>string</i>	none	Use the format detailed in “Syntax for topics” on page 16. The topic that the subscriber is requesting. Only one topic can be supplied. This is a required property, and generates an error if omitted.
streamName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	If: <ul style="list-style-type: none"> • Not null – this is the stream where the publisher publishes publications. • null – the default is SYSTEM.BROKER.DEFAULT.STREAM.
qmgrName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	This is the queue manager name used to establish the subscriber’s traditional identity. Specify it as the same value you specified when you registered the subscriber. MQ limits this string to 48 bytes.
queueName	<ul style="list-style-type: none"> • null • <i>string</i> 	null	This is the queue used to establish the subscriber’s traditional identity. Specify it as the same value you specified when you registered the subscriber. MQ limits this string to 48 bytes.
correlationAsId	<ul style="list-style-type: none"> • yes • no • generate 	no	If: <ul style="list-style-type: none"> • yes – correlationId is used as part of the subscriber’s traditional identity. You must specify correlationId, but not as 0x00. • no – correlationId is not used as part of the subscriber’s traditional identity. • generate – a system-generated correlationId is used as part of the subscriber’s traditional identity.

- Unrecognized options are ignored if you use message property. If you use message header for the msgsend or msgpublish functions, you see an error when you specify unrecognized options.
- The result of a msgsend call is a varchar string. If the message succeeds, the returned value is the message ID. If the message is not sent, the return value is null.

- In a *message_body* that is a *select_for_xml* parameter, *select_for_xml* generates a SQLX-formatted representation of the SQL result set.
- You can specify *select_for_xml* only if Adaptive Server is configured for the native XML feature. You can reference *select_for_xml* only as a scalar expression from a msgsend call.
- You must surround *select_for_xml* with parentheses, as shown in the Syntax section.
- The following restrictions apply to a runtime format for *service_provider_uri*:

```
service_provider_uri ::=
  provider_name ?destination [,user=username, password=password]
  provider_name ::= local_name | full_name
  local_name ::= identifier
  full_name ::= service_provider_class:service_provider_url
```

- The *local_name* is a provider identifier, previously registered in a call to sp_msgadmin 'register', 'provider', which is shorthand for the *full_name* specified in that call.
- The only *service_provider_class* currently supported is JMS.
- The *service_provider_url* has the form “tcp://hostname:port”. The host name can be a name or an IP address.
- A *service_provider_url* cannot have spaces.

MQ

- The status returned by msgsend is the completion status from sending the message to the specified queue. It is not the completion status from the MQ pub/sub broker. To get the completion status from the MQ pub/sub broker, specify a replyToQueue, then send a request message or request a negativeActionReport. The MQ pub/sub broker sends a response or report MQRFH message to replyToQueue. In both cases, you must explicitly read the response or report message from the replyToQueue, and check the MQPSCompCode, MQPSReason, and MQPSReasonText properties in the received message.
- When you specify *msgSegment* or *msgLastSegment*, if the application is reading the message (by specifying MQGMO_COMPLETE_MSG for a non-Adaptive Server application, or completeMsg=yes for an Adaptive Server application), all the messages making up that logical message must be sent in a unit of work, so you must send all of the messages that need to be grouped in a single transaction.

Permissions

You must have `messaging_role` to run `msgsend`.

msgsubscribe

Description	JMS only – provides a SQL interface to subscribe or unsubscribe to a topic.
Syntax	<pre>msg_subscribe ::= msgsubscribe (subscription_name) subscription_name ::= basic_character_expression</pre>
Parameters	<p><i>subscription_name</i> is the name of the subscription to which you are subscribing. A <i>basic_character_expression</i>.</p>
Examples	Tells the JMS messaging provider to begin holding messages published to the topic registered as “subscription_1”:
Usage	<ul style="list-style-type: none">• Before you specify a subscription with msgsubscribe or msguncsubscribe, you must register the subscription with sp_msgadmin. This example registers the durable subscription “subscription_1.”

```
sp_msgadmin 'register', 'subscription', 'subscription_1',
            'my_jms_provider?topic=topic.sample,user=user1,password=pwd',
            'Supplier=12345', null, 'durable1', 'client1'
```

- Once msgsubscribe is called, all messages published on the specified topic that qualify for the selector are held until msgconsume is called to read the messages. If you do not want to hold messages that arrive before you are ready to consume them, do not call msgsubscribe. Calling msgconsume without previously calling msgsubscribe starts the subscription when msgconsume is called.
- msgsubscribe starts a subscription for the client to receive messages defined by the endpoint and filter specified by *subscription_name*. It returns 0 if it succeeds, or 1 if it fails.
- If you specify with retain, the connection to the JMS messaging provider is terminated so that another subscription can connect, using the same subscriber *client_id* specified in the subscription. The durable subscriber remains defined within Adaptive Server and within the JMS message provider. If you specify with remove, the durable subscriber definition is removed from the JMS message provider. The default value is with retain.

In a separate scenario, a SQL session releases a subscription so that another session can consume messages. This example shows Session 1 releasing the subscription, so that Session 2 can begin consuming from it.

Table 3-21: SQL sessions

Session 1	Session 2
<pre>select msgunsubscribe ('subscription_1' WITH RETAIN) selectmsgconsume ('subscription_1') ... selectmsgconsume ('subscription_1') select msgunsubscribe ('subscription_1' WITH RETAIN) ...</pre>	<pre>select msgsubscribe('subscription_1') select msgconsume('subscription_1') ... select msgconsume('subscription_1') select msgunsubscribe('subscription_1' WITH RETAIN)</pre>

- The following example shows `msgsubscribe` used before the application logic is ready to read the messages that force the JMS client to hold messages. The application subscribes:

```
select msgsubscribe ('subscription_1')
```

The client consumes the message multiple times, and uses other application logic not related to messaging. It is then ready to read messages, and it receives all the messages that have arrived since `msgsubscribe` was called:

```
select msgconsume('subscription_1')
select msgconsume('subscription_1')
```

The client application is finished with this subscription, and unsubscribes:

```
select msgunsubscribe('subscription_1')
```

msgunsubscribe

Description	JMS only – provides a SQL interface to subscribe or unsubscribe to a topic.
Syntax	<pre>msg_unsubscribe::=msgunsubscribe (subscription_name [with {remove retain}]) subscription_name::=basic_character_expression</pre>
Parameters	<p><i>subscription_name</i> is the name of the subscription to which you are subscribing. A <i>basic_character_expression</i>.</p> <p>with {<i>remove</i> <i>retain</i>} removes or retains the durable subscription from the JMS message provider.</p>
Examples	<p>Tells the JMS messaging provider to stop holding messages published to the topic registered as “subscription_1”:</p> <pre>select msgunsubscribe('subscription_1')</pre>
Usage	<ul style="list-style-type: none">• Before you specify a subscription with <code>msgsubscribe</code> or <code>msgunsubscribe</code>, you must register the subscription with <code>sp_msgadmin</code>. This example registers the durable subscription “subscription_1”: <pre>sp_msgadmin 'register', 'subscription', 'subscription_1', 'my_jms_provider?topic=topic.sample,user=user1,password=pwd', 'Supplier=12345', null, 'durable1', 'client1'</pre> <ul style="list-style-type: none">• <code>msgunsubscribe</code> stops any current subscription for the client to the endpoint and filter specified by <i>subscription_name</i>. It returns a 0 if it succeeds, or 1 if it fails.• If you specify with <code>retain</code>, the connection to the JMS messaging provider is terminated so that another subscription can connect, using the same subscriber <i>client_id</i> specified in the subscription. The durable subscriber remains defined within Adaptive Server and within the JMS message provider. If you specify with <code>remove</code>, the durable subscriber definition is removed from the JMS message provider. The default value is with <code>retain</code>. <p>When you unsubscribe a subscription using with <code>remove</code>, it is possible to miss messages:</p>

```
<login>
select msgsubscribe('subscription_1')
select msgconsume('subscription_1')
```



```

...
select msgconsume('subscription_1')
select msgunsubscribe('subscription_1' WITH REMOVE)
<logout>

----Messages published to the topic registered as subscription_1 are no
----longer held by the JMS provider

<login>
select msgsubscribe('subscription_1')
select msgconsume('subscription_1')
...
select msgconsume('subscription_1')
select msgunsubscribe('subscription_1' WITH REMOVE)

```

In a separate scenario, a SQL session releases a subscription so that another session can consume messages. This example shows Session 1 releasing the subscription, so that Session 2 can begin consuming from it.

Table 3-22: SQL sessions

Session 1	Session 2
<pre> select msgunsubscribe ('subscription_1' WITH RETAIN) selectmsgconsume ('subscription_1') ... selectmsgconsume ('subscription_1') select msgunsubscribe ('subscription_1' WITH RETAIN) </pre>	<pre> select msgsubscribe('subscription_1') select msgconsume('subscription_1') ... select msgconsume('subscription_1') select msgunsubscribe('subscription_1' WITH RETAIN) </pre>

- The following example shows msgsubscribe used before the application logic is ready to read the messages that force the JMS client to hold messages. The application subscribes:

```
select msgsubscribe ('subscription_1')
```

The client consumes the message multiple times, and uses other application logic not related to messaging. Then it is ready to read messages, and it receives all the messages that have arrived since `msgsubscribe` was called:

```
select msgconsume('subscription_1')
select msgconsume('subscription_1')
```

The client application is finished with this subscription, and unsubscribes:

```
select msgunsubscribe('subscription_1')
```

endpoint

Description

MQ – specifies the general syntax and processing for *endpoint* for WebSphere MQ. Individual options are described in the functions and stored procedures that accept an *endpoint* argument.

Note JMS endpoints are opaque to Adaptive Server, and are not inspected for correctness or validity. Instead, they are sent directly to the JMS provider.

Syntax

```

service_provider_uri ::=
    provider_name?qmgr=qmgr_name,destination
provider_name ::=
    local_name | full_name
    local_name ::= identifier
    full_name ::= service_provider_class:service_provider_url
    service_provider_class ::= ibm_mq
    service_provider_url ::=
        channel_name/tcp/hostname(port)
        channel_name ::= identifier
        hostname ::= identifier
        port ::= integer
qmgr_name ::= identifier
destination ::= [remote_qmgr,]queue=queue_name
    remote_qmgr ::= remote_qmgr=remote_qmgr_name
    remote_qmgr_name ::= identifier
    queue_name ::= identifier

```

Parameters

local_name

is the name of a registered publisher or subscriber.

qmgr_name

is the name of a MQ queue manager. MQ limits the length of a queue manager name to 48 characters (bytes).

ibm_mq

defines the service provider class. It can be uppercase or lowercase.

channel_name

is the name of the MQ client channel. MQ limits the length of a channel name to 20 characters (bytes).

tcp

is the transport protocol.

hostname

is the host name of the machine where the MQ listener is running.

port

is the port number where the MQ listener is listening.

Note You cannot exceed 264 bytes in the combined length of *hostname(port)*.

queue_name

is the name of a MQ queue. MQ limits the length of a queue name to 48 characters (bytes).

remote_qmgr_name

is the name of the MQ queue manager. MQ limits the length of a queue manager name to 48 characters (bytes).

Use *remote_qmgr* when there is a remote queue. For example:

```
ibm_mq:CHANNEL2/TCP/host2(5678)?qmgr=QM2, remote_qmgr=QM3,queue=QM3.QUEUE
```

In the example:

- QM2 – is the queue manager that accepts the connection on channel “CHANNEL2.”
- QM3.QUEUE – is owned by remote queue manager QM3.
- QM2 – establishes a queue manager channel to QM3.

You must have a server-to-server channel between QM2 and QM3.

Note You must specify *qmgr*, *remote_qmgr*, and *queue_name* in that order.

The access to the MQ queue is made as the Adaptive Server login user. Unlike the JMS support, you cannot specify a user name and password with the endpoint. This means that all Adaptive Server logins that are performing messaging operations must be valid MQ users. Since MQ uses the OS user identities, the Adaptive Server login must also have a user account on the machine where the MQ queue manager is running.

Examples

Example 1 Sends the message, “hello world 1” to a local queue, which is already available on the queue manager once MQ is installed:

```
select msgsend('hello world 1',
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,
  queue=SYSTEM.DEFAULT.LOCAL.QUEUE')
```

Example 2 Sends the message, “hello world 2” to a queue:

```
select msgsend('hello world 2',  
  'ibm_mq:channel2/tcp/host2(5678)?qmgr=QM2,  
  queue=SYSTEM.DEFAULT.QUEUE')
```

Example 3 Sends the message, “hello world 3” to a queue:

```
select msgsend('hello world 3',  
  'ibm_mq:channel2/tcp/host2(5678)?qmgr=QM2,  
  remote_qmgr=QM3,queue=QM3.Q')
```

option_string

Description	Specifies the general syntax and processing for <i>option_string</i> . Individual options are described in the functions that reference them.
Syntax	<pre>option_string ::= basic_character_expression option_string_value ::= option_and_value [[,] option_and_value] option_and_value ::= option_name = option_value option_name ::= simple_identifier option_value ::= simple_identifier quoted_string integer_literal float_literal byte_literal true false null</pre>
Parameters	<p><i>option_string</i> is the string describing the option you want to specify.</p> <p><i>simple_identifier</i> is the string that identifies the value of an <i>option</i>.</p> <p><i>quoted_string</i> is the string formed using the normal SQL conventions for embedded quotation marks.</p> <p><i>integer_literal</i> is the literal specified by normal SQL conventions.</p> <p><i>float_literal</i> is the literal specified by normal SQL conventions.</p> <p>true is a Boolean literal.</p> <p>false is a Boolean literal.</p> <p>null is a null literal.</p> <p>byte_literal has the form 0xHH, where each H is a hexadecimal digit.</p>
Usage	For <i>option_string</i> usage, see <i>msgsend</i> on page 90.

sizespec

Description	MQ only – message options and property values that accept a <i>size</i> accept the following syntax as a size specification. Message options and property values that accept a size specification accept the following syntax as a size specification for MQ.
Syntax	<pre>sizespec ::= integer_number [sizespec_units] sizespec_units ::= { M K }</pre>
Parameters	<p><i>integer_number</i> is the size.</p> <p>K or k is kilobytes.</p> <p>M or m is megabytes.</p> <p><i>sizespec_units</i> is the size specification in megabytes (M) or kilobytes (K), or bytes.</p> <p>If you do not provide <i>sizespec_units</i>, the default is bytes.</p>

Examples **Example 1** shows the size specification for 100MB:

```
-- Specify buffer length to be 100 megabytes
select msgrecv('ibm_mq:channel1/tcp/host1(5678)?'
+ 'qmgr=QM1,queue=SYSTEM.DEFAULT.LOCAL.QUEUE'
option 'bufferLength=100M')
```

Example 2 shows the size specification for 300K:

```
-- Specify buffer length to be 300 kilobytes
select msgrecv(
'ibm_mq:channel2/tcp/host2(5678)?qmgr=QM2,remote_qmgr=QM3,queue=QM3.Q'
option 'bufferLength=300K')
```

Example 3 MQ – shows the size specification for 1MB:

```
-- bufferLength specified as 1 megabyte
select msgrecv(
'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=DEFAULT.QUEUE'
option 'bufferLength=1M')
```

Example 4 MQ – shows the size specification for 10K:

```
-- bufferLength specified as 10K
select msgrecv(
'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=DEFAULT.QUEUE'
option 'bufferLength=10K')
```

timespec

Description	Message options and property values that accept a time interval using the <code>timespec</code> function accept the following syntax as a time specification for both MQ and JMS.
Syntax	<pre>'timeout=<i>timespec</i>' <i>timespec</i> ::= <i>integer_number</i> [<i>timespec_units</i>] <i>timespec_units</i> ::= { <i>dd</i> <i>hh</i> <i>mi</i> <i>ss</i> <i>ms</i> }</pre>
Parameters	<p><i>dd</i> is days.</p> <p><i>hh</i> is hours.</p> <p><i>mi</i> is minutes.</p> <p><i>ss</i> is seconds.</p> <p><i>ms</i> is milliseconds.</p> <p><i>timespec_units</i> is milliseconds. If you do not provide <i>timespec_units</i>, the default is milliseconds.</p>

Examples **Example 1** Shows the time specification for 100 days:

```
-- timeout specified as 100 days
select msgrcv('ibm_mq:channel2/tcp/host2(5678)?'
  + 'qmgr=QM2,remote_qmgr=QM3,queue=QM3.Q'
  option 'timeout=100dd')
```

Example 2 Shows the time specification for 300 minutes:

```
-- timeout specified as 300 minutes
select msgrcv('ibm_mq:channel1/tcp/host1(5678)?'
  + 'qmgr=QM1,queue=SYSTEM.DEFAULT.LOCAL.QUEUE'
  option 'timeout=300mi')
```

Example 3 Shows the time specification for 1,024 milliseconds:

```
-- timeout specified as 1,024 milliseconds
select msgrcv(
  'ibm_mq:channel2/tcp/host2(5678)?'
  + 'qmgr=QM2,queue=SYSTEM.DEFAULT.LOCAL.QUEUE')
```



```
option 'timeout=1024ms')
```

Example 4 MQ – shows the time specification for 30 seconds:

```
-- timeout specified as 30 seconds
select msgrecv(
  'ibm_mq:channel1/tcp/host1(5678)?qmgr=QM1,queue=DEFAULT.QUEUE'
  option 'timespec=30ss')
```

Example 5 JMS – shows the time specification for 30 minutes:

```
-- timeout specified as 30 minutes
select msgrecv(
  'tibco)_jms:tcp://localhost:7222?queue=queue.sample'
  option 'timeout=30mi')
```

See also `msgconsume`, `msgpublish`, `msgrecv`, `msgsend`

This chapter describes sample code illustrating messaging functionality that is distributed with Adaptive Server Real-Time Data Services (RTDS).

Topic	Page
Sybase directories	139
Using code samples with Replication Server function strings	140
Using code samples with SQL	140
Using code samples with Java/JDBC	140

Sybase directories

The SYBASE directory contains three subdirectories:

- *functionstring* – scripts to generate Replication Server function strings, for converting the default SQL template into calls to the messaging system.
- *sql* – SQL scripts with samples using RTDS.
- *jdbc* – JDBC samples using RTDS.

You can find the code samples in the `$$SYBASE/$SYBASE_ASE/samples/messaging` directory.

Each subdirectory contains a *README* file, which explains the purpose of each code sample, provides a procedure for running it, and gives any installation instructions necessary.

The operating system file names in Windows and other platforms are not named exactly the same. For example, *queue_listener.bat* on a Windows platform may be simply *queue_listener* on a UNIX/Linux platform.

Using code samples with Replication Server function strings

These code samples assume that you have some basic knowledge of Replication Server setup and configuration, as well as a basic knowledge of messaging.

The code samples in `$$SYBASE/$$SYBASE_ASE/samples/messaging/functionstring` are designed to help you use Adaptive Server RepAgent™ and Replication Server for publishing database modifications, such as the commands insert, update, and delete. They also demonstrate using stored procedures as a customized message to the messaging system.

You can publish database modifications as messages without altering your application code, using the methods illustrated in these code samples. These code samples publish messages from any existing Adaptive Server (version 12.5.2 and earlier) or any non-Adaptive Server database into the message bus.

Using code samples with SQL

The code samples in `$$SYBASE/$$SYBASE_ASE/samples/messaging/sql` illustrate how you can write or modify SQL (stored procedures, triggers, and so forth), to publish customized messages to the messaging system.

These samples also illustrate how to use SQL code to consume messages from the message bus, using Adaptive Server as both a participant in messaging and as an application using the message bus.

Using code samples with Java/JDBC

The code samples in `$$SYBASE/$$SYBASE_ASE/samples/messaging/jdbc` describe how you can write or modify Java code to publish customized messages to the messaging system.

These samples also illustrate Java code that consumes messages from the message bus, using Adaptive Server as both a participant in messaging and as an application using the message bus.

Glossary

Both the JMS- and MQ-related terms defined here are used throughout this document.

broker	A WebSphere MQ process that performs subscription resolution in a pub/sub model.
channel	A WebSphere MQ object that is a logical communication link.
durable subscription	A JMS subscription that retains messages while the client is not connected.
JMS	Java Message Service.
messaging client	A JMS program that produces or consumes messages.
MOM	JMS message-oriented middleware.
MQ	WebSphere MQ Message Queue messaging system.
MQ publish/subscribe	WebSphere MQ publish-and-subscribe function.
MQI	WebSphere MQ Message Queue Interface programming API.
MQM	WebSphere MQ Message queue manager process that manages a queue.
nondurable subscription	A JMS subscription that retains messages only while the client is connected.
queue	In JMS, a domain for point-to-point messaging. In WebSphere MQ, an object that stores sent messages.
payload	A WebSphere MQ message body.
publication	In WebSphere MQ, the information that is sent by a publisher.
publisher	In WebSphere MQ, the sender in a publish/subscribe model.

RF header	The WebSphere MQ rules and formatting header used by MQ pub/sub. All messages sent to the MQ pub/sub broker or to the stream queue must have an RF header. The RF header conveys control information to the MQ pub/sub broker. In MQ pub/sub messages, the message payload contains a RF header, followed by the application data.
RFH	The WebSphere MQ rules and formatting header; the portion of the message header that provides rules and formatting information for that message
service provider	A TIBCO JMS message provider. For instance, TIBCO JMS is a service provider, called a messaging provider in this document.
stream	In WebSphere MQ, the grouping of related MQ topics.
subscriber	In WebSphere MQ, the receiver in a publish/subscribe topology.
subscription	A TIBCO JMS domain for publishing or consuming one-to-many messaging.
topic	<p>In TIBCO JMS, similar to queues in, but used for one-to-many messaging.</p> <p>In WebSphere MQ, the subject of a publication. WebSphere MQ pub/sub topics and JMS topics are different. In JMS, a topic is a pub/sub endpoint, whereas in WebSphere MQ pub/sub, a topic is a subject of a message.</p>

Note WebSphere MQ pub/sub topics and JMS topics are different. In JMS, a topic is a pub/sub endpoint, whereas in WebSphere MQ pub/sub, a topic is a subject of a message.

Index

Symbols

@@. *See* global variable.

A

Adaptive Server Enterprise, related documents for vi
Adaptive Server-specific message properties 38
alias queue (MQ) 11
ASE message types 86
ASE_MSBODY_SCHEMA message property 38
ASE_MSGBODY message property 38
ASE_ORIGIN message property 39
ASE_RTMS_CHARSET message property 38
ASE_RTMS_VERSION message property 39
ASE_SPID message property 39
ASE_TIMESTAMP message property 39
ASE_VERSION message property 39
ASE_VERSION_FORMATS message property 39
asynchronous messaging 2
audience for the book v
Availability and Certification Reports Web site xii

B

binary value of datatypes 86
broker command queue in MQ 4
broker, defined 141
built-ins. *See* functions, including individual function names.
byte message type 86
byte ordering 86

C

channel defined 141
cluster queue (MQ) 11

code samples
 using with Java/JDBC 140
 using with Replication Server function strings 140
 using with SQL 140
concepts of messaging 1
consuming messages from a JMS topic 8
conventions, syntax xiv
creating a personalized view of the Sybase Web site
 xiii
creating queues and topics 40

D

datatypes, binary value of 86
descriptions
 broker 141
 channels 141
 durable subscriptions 3
 endpoint syntax segment 131
 message-oriented middleware 1
 messaging concepts 1
 MQ 11
 MQ alias queue 11
 MQ cluster queue 11
 MQ dynamic local queue 11
 MQ dynamic model queue 11
 MQ local queue 11
 MQ message body 12
 MQ message header 12
 MQ remote queue 11
 msgconsume function 59
 msgheader XML documents 36
 msgpropcount function 62
 msgproperties XML documents 36, 37
 msgproplist function 63
 msgpropname function 65
 msgproptype function 66
 msgpropvalue function 68
 msgpublish function 70

- msgrecv** function 74
- msgsend** function 90
- msgsubscribe** function 126
- msgunsubscribe** function 128
- nondurable subscriptions 3
- option_string** syntax segment 134
- queue manager 11
- queues 141
- sizespec** syntax segment 135
- sp_config 'enable real time messaging'** stored procedure 42
- sp_engine** stored procedure 44
- sp_msgadmin** stored procedure 48
- timespec** syntax segment 136
- XML documents 36
- directories
 - functionstring* 139
 - jdbc* 139
 - sql* 139
- documentation
 - Adaptive Server Enterprise documents vi
 - EAServer documents ix
 - RepConnector documents x
 - Replication Server documents x
 - RTDS documents v
- durable subscriptions 3
 - defined 141
- dynamic local queue (MQ) 11

E

- EAServer, related documents for ix
- endpoint** syntax segment 131–133
 - described 131
 - examples 132
 - parameters 131
 - syntax 131
- examples
 - endpoint** syntax segment 132
 - global variables 35
 - messaging interface 9
 - MQ publish and subscribe process 14
 - msgconsume** function 60
 - msgheader* XML documents 37
 - msgpropcount** function 62

- msgproplist** function 63
- msgpropname** function 65
- msgproptype** function 66
- msgpropvalue** function 68
- msgpublish** function 71
- msgrecv** function 75
- msgsend** function 94–98
- msgsubscribe** function 126
- msgunsubscribe** function 128
- sizespec** syntax segment 135
- sp_configure 'enable real time messaging'** stored procedure 42
- sp_engine** stored procedure 45
- sp_msgadmin** stored procedure 54
- timespec** syntax segment 136
- XML documents 37

F

- figures
 - flow of MQ publication/subscription process 15
- file names, different on different platforms 139
- finding
 - latest information on component certifications xii
 - latest information on EBFs and software maintenance xiii
 - latest information on product certifications xii
- font conventions xiv
- functions
 - described 40
 - list of 40
 - list of functions 40
 - msgconsume** 59–61
 - msgpropcount** 62
 - msgproplist** 63–64
 - msgpropname** 65
 - msgproptype** 66–67
 - msgpropvalue** 68–69
 - msgpublish** 70–73
 - msgrecv** 74–89
 - msgsend** 90–125
 - msgsubscribe** 126
 - msgunsubscribe** 128
 - rtrim** for removing trailing blanks 36

functionstring subdirectory in *\$SYBASE* directory
139

G

global variables

- `@@msgcorrelation` 30
- `@@msgcreplyqmgr` 34
- `@@msgheader` 30
- `@@msgid` 33
- `@@msgmsgschema` 34
- `@@msgproperties` 33
- `@@msgreplytoinfo` 34
- `@@msgstatus` 34
- `@@msgstatusinfo` 35
- `@@msgtimestamp` 35
- examples 35
- setting 30
- usages 36

H

help, for installation or feature xv

I

IBM WebSphere MQ. *See* MQ.

J

Java Message Service. *See* JMS.

Java Technologies Web site xi

Java/JDBC, using code samples with 140

jdbc subdirectory in *\$SYBASE* directory 139

JMS

- defined 141
- message bus 2
- message properties 5
- msgrecv** *option_string* values 86
- msgrecv** *property_option_clause* values 111
- msgsend** *option_string* values 99
- queue description 3

queue, messages read from 8

reference documents xi

URL for xi

JMS topics

publishing and consuming messages 8

K

keywords 39

message header 39

message property 39

message selector 39

transactional messaging full 39

transactional messaging none 39

transactional messaging simple 39

with remove 39

with retain 39

L

local queue (MQ) 11

M

message

body 4

bus, TIBCO 2

formats 4

headers 4

interface, preview of 9

properties in JMS 5

properties in MQ 5

properties, working with 8

read from JMS queue 8

selectors in JMS 5

selectors in MQ 5

types in MQ 12

message filters for using **msgrecv** function 88

message formats 4

message header keyword 39

message headers 4

message message type 86

message properties 8

Index

- Adaptive Server-specific 38
- ASE_MSBODY_SCHEMA 38
- ASE_MSGBODY 38
- ASE_ORIGIN 39
- ASE_RTMS_CHARSET 38
- ASE_RTMS_VERSION 39
- ASE_SPID 39
- ASE_TIMESTAMP 39
- ASE_VERSION 39
- ASE_VERSION_FORMATS 39
- JMS, in 5
- MQ, in 5
- See also* **msgsend** function 8
- message property** keyword 39
- Message Queue Interface (MQI) described 11
- message receivers in JMS 3
- message selector** keyword 39
- message selectors 5
- message senders in JMS 3
- message types
 - binary 7
 - supported in **msgconsume** 61
 - text 7
- message-oriented middleware (MOM) 141
- message-related global variables 30
- messages
 - publishing and consuming from a topic 8
 - sending and receiving from a queue 7
 - sending with Transact SQL applications 7
- messaging
 - client 141
 - concepts 1
 - models 3, 4
- messaging global variables
 - @@msgcorrelation** 30
 - @@msgheader** 30
 - @@msgid** 33
 - @@msgmsgschema** 34
 - @@msgproperties** 33
 - @@msgreplyqmgr** 34
 - @@msgreplyto**, format 36
 - @@msgreplytoinfo** 34
 - @@msgstatus** 34
 - @@msgstatusinfo** 35
 - @@msgtimestamp** 35
 - char datatypes 36
- messaging grouping in MQ 13
- messaging models
 - JMS 3
 - MQ 4
 - MQ publish and subscribe 4
 - MQSeries-defined 4
 - point-to-point 3
 - publish and subscribe 3
- messaging provider 2
 - creating, deleting, and accessing queues and topics 40
- messaging systems, asynchronous 2
- model queue (MQ) 11
- models, messaging 3, 4
- MOM. *See* message-oriented middleware
- MQ
 - alias queue described 11
 - broker command queue 4
 - cluster queue described 11
 - dynamic local queue described 11
 - local queue described 11
 - message body described 12
 - message grouping 13
 - message header described 12
 - message properties 5
 - Message Queue Interface (MQI) 11
 - message types 12
 - messaging models 4
 - model queue described 11
 - msgrecv** function, usage for 87
 - msgrecv** *option* and *option_string* values 78
 - msgsend** *option_string* values 100
 - msgsend** properties if **rfhCommand** is set to **deletePublications** 112
 - msgsend** *property_option_clause* values 102
 - overview 11
 - publish and subscribe description 13
 - publish and subscribe examples 18
 - publish and subscribe messaging model 4
 - publish and subscribe process examples 14
 - publisher and subscriber identities 18
 - remote queue described 11
 - RF headers 5
 - syntax for topics 16
 - URL xi
- MQI (Message Queue Interface) 11

- description 11
- msgconsume** function 8, 59–61
 - calling, results of 61
 - described 59
 - examples 60
 - message types supported 61
 - parameters 59
 - syntax 59
 - unsupported message datatypes 61
 - usage 61
- @@msgcorrelation** messaging global variable 30
- @@msgheader** messaging global variable 30
- msgheader** XML document 36–38
 - described 36
 - examples 37
 - syntax 37
 - usage 38
- @@msgid** messaging global variable 33
- msgpropcount** function 9, 62
 - described 62
 - examples 62
 - parameters 62
 - syntax 62
- @@msgproperties** messaging global variable 33
- msgproperties** XML document 36–38
 - described 36, 37
 - syntax 37
 - usage 38
- msgproplist** function 9, 63–64
 - described 63
 - examples 63
 - parameters 63
 - syntax 63
 - usage 63
- msgpropname** function 9, 65
 - described 65
 - examples 65
 - parameters 65
 - syntax 65
- msgproptype** function 9, 66–67
 - described 66
 - examples 66
 - parameters 66
 - syntax 66
 - usage 67
- msgpropvalue** function 9, 68–69
 - described 68
 - examples 68
 - parameters 68
 - syntax 68
- msgpublish** function 8, 70–73
 - described 70
 - examples 71
 - option_string* values 72
 - parameters 70–71
 - properties_clause* values 72
 - syntax 70
 - usage 71–72
- msgpublish** *option_string* values 72
- msgpublish** *properties_clause* values 72
- msgrecv** function 74–89
 - described 74
 - examples 75
 - JMS *option* and *option_string* values 86
 - message filters 88
 - MQ *option* and *option_string* values 78
 - parameters 74
 - permissions 89
 - syntax 74
 - usage 86
 - usage for MQ 87
- @@msgreplyqmgr** messaging global variable 34
- @@msgreplytoinfo** messaging global variable 34
- @@msgschema** messaging global variable 34
- msgsend** function 90–125
 - behavior in a transactions 25
 - described 90
 - examples 94–98
 - JMS *option_string* values 99
 - JMS *property_option_clause* values 111
 - MQ *option_string* values 100
 - MQ *property_option_clause* values 102
 - msgsend** properties if **rflCommand** is set to **deletePublications** 112
 - parameters 90–94
 - permissions 125
 - syntax 90
 - usage 98–124
- @@msgstatus** messaging global variable 34
- @@msgstatusinfo** messaging global variable 35
- msgsubscribe** function 8, 126
 - described 126

Index

- examples 126
- parameters 126
- syntax 126
- usage 126

@*msgtimestamp* messaging global variable 35

msgunsubscribe function 8, 128

- described 128
- examples 128
- parameters 128
- syntax 128
- usage 128

N

nondurable subscriptions 3, 141

O

option strings 40

option_string syntax segment 134

- described 134
- parameters 134
- syntax 134
- usage 134

P

parameters

- endpoint** syntax segment 131
- msgconsume** function 59
- msgpropcount** function 62
- msgproplist** function 63
- msgpropname** function 65
- msgproptype** function 66
- msgpropvalue** function 68
- msgpublish** function 70–71
- msgrecv** function 74
- msgsend** function 90–94
- msgsubscribe** function 126
- msgunsubscribe** function 128
- option_string** syntax segment 134
- sizespec** syntax segment 135
- sp_config 'enable real time messaging'** stored

- procedure 42
- sp_engine** stored procedure 44
- sp_msgadmin** stored procedure 49
- timespec** syntax segment 136

performing messaging operations described 2

permissions

- msgrecv** function 89
- msgsend** function 125
- sp_engine** stored procedure 47
- sp_msgadmin** stored procedure 57

point-to-point messaging models 3

- JMS 3
- MQSeries 4

point-to-point queues

- JMS 3
- MQ 4

preview, examples 9

previewing message interface 9

procedures

- creating a personalized view of the Sybase Web site xiii
- finding the latest information on component certifications xii
- finding the latest information on EBFs and software maintenance xiii
- finding the latest information on product certifications xii

Product Manuals Web site vi

provider, messaging 2

publish and subscribe

- described 13
- examples 18
- messaging model 3

publish-and-subscribe

- JMS messaging model 3
- MQ messaging model 4

publisher and subscriber identities in MQ 18

publishing messages from a JMS topic 8

Q

queue manager description 11

queues

- defined 141
- for one-to-one messaging 141

sending and receiving messages from 7
 queues and topics, creating, deleting, accessing 40

R

receiving messages 7
 referenced documents xi
 TIBCO EMS xi
 WebSphere MQ xi
 related documents
 Adaptive Server Enterprise vi
 EAServer ix
 RepConnector x
 Replication Server x
 remote queue (MQ) 11
 RepConnector 2
 related documents for x
 Replication Server
 code samples 140
 related documents for x
requeue, using for incorrect message types 86
 RF headers 5
 RFH. *See* RF headers.
rtrim function 36
 rules and formatting headers. *See* RF headers.

S

sample code
 overview 139
 sybase directories 139
 samples 139
 sending messages 7
 service provider 142
set transactional messaging command 25
sizespec syntax segment 135
 described 135
 examples 135
 parameters 135
 syntax 135
sp_config 'enable real time messaging' stored
 procedure 42–43
 described 42
 parameters 42

 syntax 42
 usage 42
sp_configure 'enable real time messaging' stored
 procedure
 examples 42
sp_engine stored procedure 44–47
 described 44
 examples 45
 parameters 44
 permissions 47
 syntax 44
 usage 46
sp_msgadmin and MQSeries 40
sp_msgadmin stored procedure 8, 48–58
 described 48
 examples 54
 parameters 49
 permissions 57
 syntax 48
 usage 56
 SQL
 commands in a transaction 25
 functions described 40
 functions with message properties 8
 using code samples with 140
sql subdirectory in *\$\$SYBASE* directory 139
 stored procedures
 list of 40
 sp_config 'enable real time messaging' 42–43
 sp_engine 44–47
 sp_msgadmin 48–58
 style conventions xiii
 subscriptions
 defined 142
 durable 3
 nondurable 3
 support contracts with Sybase xv
 Sybase product certifications Web site xii
 Sybase Product Manuals Web site vi
 Sybase Support Web site xiii
 Sybase Technical Support xv
 syntax
 endpoint syntax segment 131
 msgconsume function 59
 msgheader XML documents 37
 msgpropcount function 62

msgproperties XML documents 37
msgproplist function 63
msgpropname function 65
msgproptype function 66
msgpropvalue function 68
msgpublish function 70
msgrecv function 74
msgsubscribe function 126
msgunsubscribe function 128
option_string syntax segment 134
sizespec syntax segment 135
sp_config 'enable real time messaging' stored procedure 42
sp_engine stored procedure 44
sp_msgadmin stored procedure 48
timespec syntax segment 136
syntax conventions xiv
syntax for topics in MQ 16
syntax segments
 endpoint 131–133
 list of 41
 option_string 134
 sizespec 135
 timespec 136

T

tables
 @*msgheader* global variable fields and descriptions 30
 Adaptive Server-specific message properties 38
 msgconsume *option* and *option_string* parameter values 59
 msgpublish *option_string* values 72
 msgpublish *properties_clause* values 72
 msgrecv *option* and *option_string* values for MQ 78
 msgrecv *option_string* values for JMS 86
 msgrecv *property_option_clause* values for JMS 111
 msgsend *option_string* values for JMS 99
 msgsend *option_string* values for MQ 100
 msgsend properties if **rfhCommand** is set to **deletePublications** for MQ 112
 msgsend *property_option_clause* values for MQ 102
technical support xv
text message type 86

TIBCO EMS
 message bus 2
 URL xi
timespec syntax segment 136
 described 136
 examples 136
 parameters 136
 syntax 136
topics
 defined 142
 publishing and consuming messages from 8
 trailing blanks, removing with **rtrim** 36
 transactional behavior, controlling with **set transactional messaging** 25
transactional messaging full keyword 39
transactional messaging none keyword 39
transactional messaging simple keyword 39
transactions
 committing 25
 database, effect on messages 25
 SQL commands 25
Transact-SQL, sending messages with 7

U

URLs
 Availability and Certification Reports xii
 IBM WebSphere MQ xi
 Java at Sun xi
 Java Technologies xi
 Sybase product certifications xii
 Sybase product manuals vi
 Sybase Support xiii
 TIBCO EMS xi
 WebSphere MQ xi
usages
 global variables 36
 msgconsume function 61
 msgheader XML documents 38
 msgproperties XML documents 38
 msgproplist function 63
 msgproptype function 67
 msgpublish function 71–72
 msgrecv function 86
 msgsend function 98–124

msgsend syntax 90
msgsubscribe function 126
msgunsubscribe function 128
option_string syntax segment 134
sp_config 'enable real time messaging' stored procedure 42
sp_engine stored procedure 46
sp_msgadmin stored procedure 56
XML documents 38

W

WebSphere MQ
 reference documents xi
 See also MQ
 URL xi
with remove keyword 39
with retain keyword 39

X

XML documents
 description 36
 examples 37
 msgheader 36–38
 msgproperties 36–38
 usage 38

