ADMINISTRATION GUIDE | PUBLIC
SAP Adaptive Server Enterprise 16.0 SP03
Document Version: 1.0 – 2020-03-04

Active Messaging Users Guide

THE BEST RUN SAP

Content

1 Introduction to Active Messaging

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

Many SAP ASE customers use messaging and queuing, or publish-and-subscribe systems in their own application environments. These applications are called message-oriented middleware. Often, the same application combines database operations with messaging operations.

1.1 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 when they occur, rather than simply queuing them.

An event, such as a change in a record, must be evaluated 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 Active Messaging using the Tibco or EAServer JMS message bus, or IBM WebSphere MQ.

1.2 JMS Messaging Models

JMS messaging models include publish-and-subscribe (topics) and point-to-point (queues).

The publish-and-subscribe (pub/sub) model is a one-to-many 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 the pub/sub model:

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

The point-to-point model is a one-to-one model, in 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.

1.3 WebSphere MQ

All WebSphere MQ messaging models are point-to-point; where 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.

1.4 Message Format

The message format for both MQ and JMS consists of a message header and message body.

A 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.

A 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.

Type	Description
JMS Message Properties	In Tibco, EAServer, and Sonic MQ, message properties are user-defined properties that you can include with the message. Message properties have types that 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	With 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.

1.5 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.

2 Understanding Active Messaging

Active Messaging allows you to use SAP ASE as a client of the message provider. You can use Transact-SQL to send messages to or retrieve messages from the messaging provider.

2.1 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.

To construct a message body or payload, use application logic. You may constuct the body from 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 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.

2.2 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.

Before you can use `msgpublish`, `msgconsume`, and `msgsubscribe`, register a subscription using `sp_msgadmin 'register'`.

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

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.

2.3 Working with Message Properties

When a message is read, Transact-SQL application logic can process the message header and user properties by 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`

2.4 Examples of Previewing the Messaging Interface

Examples providing a preview of the Transact-SQL messaging interface.

Example: 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: 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: Example 3

(JMS) Publishes a message to a topic:

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

Example: Example 4

(JMS) Consumes a message from a topic:

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

Example: 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: Example 6

(MQ) sends a message to a queue:

```
select msgsend('hello world',
       'ibm_mq:channel1/tcp/host1(1234)?qmgr=QM,queue=DEFAULT.QUEUE'
       message header 'priority=2')
```

Example: Example 7

(MQ) Reads a message from a queue:

```
select msgrecv(
       'ibm_mq:channel1/tcp/host1(1234)?qmgr=QM,queue=DEFAULT.QUEUE'
       option 'timeout=30ss')
```

Example: Example 8

(MQ) Registers a publisher and publishes a message about fish:

```
select msgsend(NULL,
    'ibm_mq:channel1/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:channel1/tcp/host1(1234)?qmgr=QM,queue=ANIMALS.STREAM'
    message header 'topics=fish')
```

Example: Example 9

(MQ) Registers a subscriber, reads a message, and processes the message properties:

```
select msgsend(NULL,
     'ibm_mq:channel1/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:channel1/tcp/host1(1234)?qmgr=QM,queue=MY_ANIMALS.QUEUE'
       option 'timeout=30ss')
select msgpropvalue('MPQScompcode', @@msgproperties)
```

2.5 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.

2.5.1 MQ Queue Objects

A queue manager is a system program that provides queuing services, and owns and manages the set of resources such as queues, channels, and process definitions, that are used by WebSphere MQ. A queue is a data structure used to store messages.

WebSphere MQ has several types of queue objects:

Type	Description
Local queue object	Identifies a local queue belonging to the queue manager to which the application is connected. All queues are local queues in that each queue belongs to a queue manager, and for that queue manager, the queue is a local queue.
Remote queue object	Identifies a queue belonging to another queue manager that is a different queue manager from the one to which the application is connected. This queue must be defined as a local queue to the queue manager to which the remote queue object belongs.
Alias queue object	This type is not a queue, but an object pointer to a local or remote queue.
Model queue object	This type defines a set of queue attributes that is used as a template to create a dynamic queue.

All types of queue objects can be sent in messages, but messages can be read only from local queue objects.

2.5.2 MQ Queue Concepts

In addition to the queue object types that are available in WebSphere MQ, there are additional concepts about queues as well.

Concept	Description
Remote queue definitions	Definitions for queues that are owned by another queue manager, and not queues themselves. Remote queue definitions enable an application to put a message to a remote queue without having to specify the name of the remote queue or the remote queue manager, or the name of the transmission queue.
Predefined queues	Created by an administrator using the appropriate MQ Series commands (MQSC) or WebSphere MQ programmable command format (PCF) commands. Predefined queues are permanent, existing independently of the applications that use them, and persisting through WebSphere MQ restarts.
Dynamic queues	Created when an application issues an MQOPEN request specifying the name of a model queue. The queue created is based on a template queue definition, which is called a model queue. The attributes of dynamic queues are inherited from the model queue from which they are created.

Concept	Description
Cluster queue objects	Hosted by a cluster queue manager and are made available to other queue managers in the cluster.

2.5.3 MQ Channels

A channel is a logical communication link between a WebSphere MQ client and a WebSphere MQ server, or between two WebSphere MQ servers.

There are two categories of channels in WebSphere MQ.

Channel	Description
Message channels	One-way links that connect two queue managers via message channel agents.
MQI channels	Connect a WebSphere MQ client to a queue manager on a server machine, and are established when you issue an MQCONN or MQCONNX call. An MQ channel is a two-way link used to transfer only MQI calls and responses. There are two channel types for MQI channel definitions: • Client-connection channel – connects to the WebSphere MQ client. • Server-connection channel – connects to the server running the queue manager, which communicates with the WebSphere MQ application that is running in an WebSphere MQ client environment.

The MQ channel supports the industry-standard Secure Sockets Layer (SSL) protocol. See your WebSphere MQ documentation from IBM for information on whether SSL is available on your platform in version 5.3 or 6.0 of MQ.

2.5.4 MQ Messages

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.
- 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.

2.6 Securing Channels with SSL

Send and receive messages through SSL.

To send and receive messages through SSL:

1. Create a key repository for the connected queue manager that contains queue manager's private key, and the digital certificate for SAP ASE.
2. Create a key repository for SAP ASE that contains the digital certificate for that SAP ASE, as well as for the connected queue managers.
3. Create an SSL-enabled server connection channel on the connected queue manager.
   Configure your key repository for SAP ASE by using the sp_msgadmin 'config', 'ibmmq_keystore' stored procedure described in `sp_msgadmin`.

In this sample scenario, WebSphere MQ communicates both with and without SSL in Active Messaging.

There are two server connection channels on queue manager "BACH"; the first, "CH1", is a normal connection, while "CH2" is configured to require SSL. The SSL cipher specification for the channel is NULL_MD5.

1. Send a message to the queue manager without enabling SSL:

```
select msgsend('a', "ibm_mq:CH1/tcp/host1(7654)?qmgr=BACH,queue=Q1')
```

2. Next, send a message to the queue manager using the SSL protocol:
   1. Set up the key repositories for the queue manager and SAP ASE separately. The key database file for SAP ASE is `/var/mqm/clients/ssl/ASE.kdb`. To set up key repositories, see your WebSphere MQ documentation from IBM.
3. Configure the key repository for SAP ASE:

```
sp_msgadmin 'config', 'ibmmq_keystore', '/var/mqm/clients/ssl/ASE'
```

4. Send the message through SSL:

```
select msgsend('e', 'ibm_mq:CH2(ssl:sslciph=NULL_MD5)
   /tcp/host1(7654)?qmgr=BACH,queue=Q1')
```

2.7 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 such as subscriber registration and cancellation to the pub/sub broker.
- 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.

More queues may 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*.

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:channel1/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')
```

3. 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')
```

4. Subscriber reads the forwarded message from MY_ANIMALS:

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

A message can have one or more topics. The WebSphere MQ pub/sub model 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. 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*.

2.7.1 Publisher and Subscriber Identities

By default, a publisher or subscriber identity consists of a queue name, a queue manager name, and optionally a correlation identifier.

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, allowing several applications to share a queueor allowing a single application to differentiate publications originating from different subscriptions.

2.7.2 MQ Publish and Subscribe Examples

Publisher and subscribe examples for MQ.

Publisher Example

The SAP ASE 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 SAP ASE 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 msgrecv(@QM + ',queue=' + @SUBQUEUE option 'timeout=30ss')
------------------------------------------------------------------------
publication on topicA
-- Get the second message on the subscriber queue, it will be on topicA.
select msgrecv(@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 you how to 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
```

2.8 Working with MQ Cluster Queue Objects

Active Messaging allows you to use SAP ASE as a client to communicate with the WebSphere MQ cluster feature.

Use `msgsend` to send messages to all the cluster queues on any cluster that is connected to a queue manager.

> **i** Note
>
> The `msgrecv` function does not support remote queue connections.

A cluster can have more than one queue manager hosting an instance of the same queue. For example, two queue managers, named MASTER_MQ1 and SLAVE_MQ1, both host cluster queue CQ1. Both queue managers then join cluster INV_CQ1, resulting in two instances of the CQ1 cluster queue in the cluster INV_CQ1.

To specify your remote queue manager, use `remote_qmgr` in your endpoint syntax segment. Ignore this `remote_qmgr` option if you are sending a message to the cluster queue that holds multiple instances, and you do not care which instance the destination is or do not need to balance the workload between cluster queue instances. In such cases, WebSphere MQ balances the workload on its own:

- If there is a instance on the connected queue manager, WebSphere MQ automatically chooses it.
- If there is no instance on the connected queue manager, WebSphere MQ determines which instance is suitable.

If you prefer not to use the default algorithm, define a cluster workload exit. An exit is a feature of WebSphere MQ that is similar to a trigger in a database. For more information on exits and how to define them, see your IBM WebSphere MQ documentation.

By using clusters with multiple instances of the same queue, you can route a message to any queue manager that hosts a copy of the correct queue. However, this adversely affects users who have multiple messages that need to maintain their sequential integrity. For example, a customer sends the following messages to a vendor:

1. "Send 100 widgets," sent at 9:00 a.m.
2. "Send 50 widgets," sent at 9:30 a.m.
3. "Cancel the first request," sent at 10:00 a.m.

In this example, the messages must maintain the correct sequence for the vendor to know that the final quantity the customer wishes to purchase is 50 widgets (that is, 100 + 50 – 100 = 50). If message 2 were to arrive before message 1, the vendor would erroneously believe the customer wished to purchase 100 widgets.

Users can address this issue by putting these messages in the same instance by specifying `clustQBinding`, an `option_string` type in the `msgsend` function. The options for `clustQBinding` are `bind`, `nobind`, and `default`.

2.9 Working with Remote Queue Objects

Transfer messages between queue managers.

Send messages to remote queue objects by using the `msgsend remote_qmgr` option to specify the names of your remote queue managers when:

- The local queue manager and the remote queue manager are in a single cluster, and the local queue manager stores the cluster queue manager definition of the remote queue manager.
- There is a transmit queue on the local queue manager, and the name of the transmit queue is the same as the one on the remote queue manager.
- There is a queue manager alias on the local queue manager, and the name of the queue manager alias is the same as the one on the remote queue manager.

> **i Note**
>
> SAP ASE sets the remote queue manager as the target queue manager, and the queue as the target queue. As soon as SAP ASE sends a message to the related transmit queue, SAP ASE returns with successful status, even though it has not yet sent a message to the target queue.

For more information on how WebSphere MQ transfers messages between queue managers, see your IBM documentation.

After a message is placed in the transmit queue, the local queue manager looks for the remote queue manager definition in its own subnet. If the local queue manager is:

- The full repository of the cluster – the local queue manager should contain a definition for the remote cluster queue manager.
- A partial repository – the local queue manager might not know where the remote definition is, in which case WebSphere MQ returns an error. When this happens, however, the local queue manager does not then ask for the location of the remote queue manager.
  If the local queue manager finds the remote queue manager definition, the local queue manager sends a message to the remote queue manager through the cluster transmit queue, after which the remote queue manager sends a message to the target queue. This way, the operation succeeds even though the target queue is not a cluster queue.

For other circumstances, the channel of the related transmit queue receives messages and sends them to the queue manager that the channel connects to. If no such channel exists or the channel has not been started, the transmit queue stores the messages until the channel is started.

2.10 Working with Text Messaging

Both JMS and WebSphere MQ can handle byte messages and text messages.

2.10.1 Text Messages and JMS

When sending or receiving messages in JMS, Active Messaging automatically detects the datatype of the message payload and handles it appropriately as either a byte or text message.

When sending messages, JMS recognizes `char`, `varchar`, `unichar`, `univarchar`, `text`, and `unitext` as valid text message types.

Example

Sends a text message to the JMS messaging bus:

```
declare @msg varchar(1024)
    select @msg = 'abcd'
    select msgsend(@msg,
        'tibco_jms:tcp://my_jms:7222?queue=sample,user=admin')
```

Example

Receives a text message from JMS messaging bus:

```
select msgrecv('tibco_jms:tcp://my_jms:7222?
    queue=sample,user=admin', returns varchar(1024))
```

Example

Sends a byte message to JMS messaging bus:

```
declare @msg varbinary(1024)
    select @msg = 'abcd'
```

```
    select msgsend(@msg,
        'tibco_jms:tcp://my_jms:7222?queue=sample,user=admin')
```

Example

Example 4 Receives a byte message from JMS messaging bus:

```
select msgrecv('tibco_jms:tcp://my_jms:7222?
    queue=sample,user=admin', returns varbinary(1024))
```

2.10.2 Text Messages and MQ

When receiving messages in WebSphere MQ, MQ regards the message as a text message only if the `formatName` message property is set to `MQSTR`. Otherwise, MQ handles the message as a byte message.

Example

Sends a text message to WebSphere MQ.

```
declare @msg varchar(1024)
    select @msg = 'abc'
    select msgsend(@msg,'ibm_mq:channel1/TCP/host1(7654)?
    qmgr=QM,queue=Q1,alter_user=yes',message property "formatName=MQSTR")
```

Example

Receives a text message from WebSphere MQ:

```
select msgrecv('ibm_mq:channel1/TCP/host1(7654)?
    qmgr=QM,queue=Q1,alter_user=yes',
    option 'bufferLength=20000k,timeout=60000',
    returns varchar(1024))
```

Example

Sends a byte message to WebSphere MQ:

```
declare @msg varbinary(1024)
    select @msg = 'abc'
    select msgsend(@msg,'ibm_mq:channel1/TCP/host1(7654)?
```

```
        qmgr=QM,queue=Q1,alter_user=yes')
```

Example

Receives a byte message from WebSphere MQ:

```
select msgrecv('ibm_mq:channel1/TCP/host1(7654)?
    qmgr=QM,queue=Q1,alter_user=yes',
    option 'bufferLength=20000k,timeout=60000',
    returns varbinary(1024))
```

Example

You can send a byte payload as a text message in WebSphere MQ as long as the payload is UTF8-encoded. In this example, text message “abc” is based on byte payload 0x616263 because the UTF8 encoding of text “abc” is 0x616263:

```
declare @msg varbinary(1024)
    select @msg = 0x616263
    select msgsend(@msg,'ibm_mq:channel1/TCP/host1(7654)?
    qmgr=QM,queue=Q1,alter_user=yes',
    message property "formatName=MQSTR")
```

2.11 SAP ASE Cluster Edition Support

Active Messaging supports client technologies, login redirection, and extended high availability of the SAP ASE Cluster Edition.

Login redirection – the ability of an instance to redirect an incoming client connection to another instance prior to acknowledging the login. Login redirection occurs during the login sequence. The client application does not receive notification that it was redirected.

Extended high availability – in an extended failover configuration, SAP ASE provides a list of failover addresses to “high-availability-aware” clients when they connect. This allows high-availability-aware clients or applications to fail over multiple times if the instance to which they are connected fails.

These clients are not required to have a HAFAILOVER entry in their interfaces file or directory services. However, if they do have an HAFAILOVER entry in their interfaces file or directory services, the clients continue to use this entry until SAP ASE sends them a list of failover addresses or servers to connect to. The clients always use the latest list SAP ASE provides.

> **i** Note
>
> The Active Messaging feature does not support connection migration, which occurs when an existing client is transferred from one instance of a cluster to another.

2.11.1 Login Redirection

Login redirection is used by the SAP ASE workload manager to send incoming connections to specific instances based on the logical cluster configuration and the cluster's current workload.

Login redirection occurs at login time when an instance tells a client to log in to another instance because of load considerations.

You need not perform any additional configuration for client redirection; it occurs automatically.

This example includes the instances "ase1" and "ase2" on nodes "blade1" and "blade2" running in the cluster "mycluster."

```
ase1
   query tcp ether blade1 19786
ase2
   query tcp ether blade2 19786
mycluster
   query tcp ether blade1 19786
   query tcp ether blade2 19786
```

For example, if Active Messaging is enabled on "ase1" and "ase2," and an application server tries to connect to "ase1" but "ase1" is unavailable, this login redirects to the "ase2" instance to perform the messaging operation.

2.11.2 Extended High Availability

SAP ASE provides a list of failover addresses to "HA-aware" clients when they connect. This allows high-availability-aware clients or applications to fail over multiple times, whenever the instance to which it is connected becomes unavailable.

If the instance has not sent a failover list to the client, the client uses the HAFAILOVER entry information in the interfaces file.

This example allows an HA-aware client to fail over if there is a network failure during login before the instance sends the extended high-availability list:

```
ase1
   query tcp ether blade1 19786ase2
   query tcp ether blade2 19786
mycluster
   query tcp ether blade1 19786
   query tcp ether blade2 19786
   hafailover mycluster
```

The HAFAILOVER entry should use the cluster alias as the server name since a client application tries each query line until it establishes a connection to an instance in the cluster. See the Clusters Users Guide for information on how to enable extended high-availability in a cluster environment.

Open Client uses the CS_PROP_EXTENDEDFAILOVER property for extended failover.

2.12 Active Messaging Support for the Threaded Kernel

Certain parameters should not be configured if you are using Active Messaging with the threaded kernel.

Do not configure these parameters if you are using Active Messaging with the threaded kernel:

- `max online Q engines`
- `number of Q engines at startup`
- `max online engines`
- `number of engines at startup`

Configure `syb_blocking_pool` to have at least one thread, and, for performance reasons, be at least as high as the SAP ASE sessions running Active Messaging. For example, if the thread count for `syb_blocking_pool` prior to configuring Active Messaging is 2, after you configure Active Messaging, three additional SAP ASE sessions use Active Messaging. In such a scenario, configure the thread count for `syb_blocking_pool` to 5 because 2+3=5.

2.13 Internationalization Support

Internationalization between SAP ASE and the messaging bus for both sending and receiving messages is supported.

The following are supported:

- The sender's server character set is configured to use GB18030 (simplified Chinese) – the sender can send a Chinese message to the messaging bus.
- The receiver's server character set is configured to use Big5 (traditional Chinese) – the receiver can receive the Chinese message from the messaging bus.

Example

This example sets the current character set, then sends a Chinese word to messaging bus in one server using the GB18030 character set:

```
1> sp_configure "default character set id"
2> go
Parameter Name       Default Memory Used Config Value Run Value Unit  Type
--------------       --------------- ----------- ---------- --------- ----
default character set id  1            0          173       173  id  static
(1 row affected)
(return status = 0)
1> declare @msg varchar(1024)
2> select @msg = 0xd6d0cec4
3> select msgsend(@msg,'ibm_mq:channel1/TCP/host1(7654)?
   qmgr=QM,queue=Q1,alter_user=yes',message property "formatName=MQSTR")
```

Example

This example receives the Chinese message from messaging bus in another server, which is running the Big5 character set:

```
1> sp_configure "default character set id"
2> go
Parameter Name       Default  Memory Used  Config Value  Run Value  Unit  Type
-------------- --------- ----------- ------------ ------------ ----- -------
default character set id   1            0           161        161 id static
(return status = 0)
1> declare @msg varchar(1024)
2> select @msg = msgrecv('ibm_mq:channel1/TCP/host1(7654)?
   qmgr=QM,queue=Q1,alter_user=yes',
3> option 'bufferLength=100k,timeout=60000',
4> returns varchar(16384))
5> select convert(varbinary(1024), @msg)
6> go
------------------------------------------------------------------------------
0xa4a4a4e5
```

The output, "0xa4a4a4e5," is the binary representation of the word "CHINESE" in the Chinese language in the Big5 character set.

2.14 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.

2.15 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 SAP ASE process is running as.

After making the connection to the MQ queue manager, SAP ASE then attempts to open the queue as the SAP ASE login that is performing the operation. For this reason, the user must:

- The user must have a user account on the machine on which the MQ queue manager is running. Without such an account, the user must use the `msgsend` function's `alter_user=yes` option to perform messaging operations.

- The user must have MQ authorizations.

> **i Note**
>
> The SAP ASE messaging_role is still required to execute Active Messaging built-in functions.

In addition, the `register, login` and `default, login` functions of `sp_msgadmin` do not allow you to register SAP ASE logins, or to create default SAP ASE logins if the endpoint specified is a queue manager.

2.16 Installing MQ Client on Host Machines

Install the MQ client software on the SAP ASE host machine.

SAP ASE dynamically loads the MQ client shared libraries.

This table shows where to install the shared libraries.

Table 1: MQ client shared libraries and directories

Platform	Directory	Library name
Solaris 32	`/opt/mqm/lib`	`libmqmcs.so, libmqic.so`
Solaris 64	`/opt/mqm/lib64`	`libmqmcs.so, libmqic.so`
Solaris AMD64	`/opt/mqm/lib64`	`libmqmcs.so, libmqic.so`
Linux 32	`/opt/mqm/lib`	`libmqic_r.so`
Linux AMD64	`/opt/mqm/lib64`	`libmqic_r.so`
HPPA 64	`/opt/mqm/lib64`	`libmqic.sl`
HPIA 64	`/opt/mqm/lib64`	`libmqic.so`
AIX 64	`/usr/mqm/lib64`	`libmqic_r.a(mqic_r.o)`
Windows 32	`c:\Program Files\IBM \Websphere MQ\bin`	`MQIC32.DLL`

- HP, HPIA, Linux, Linux AMD, Solaris, and Solaris AMD – SAP ASE 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` before you start SAP ASE.
- Windows – set %PATH% to include the library before you start SAP ASE.

2.17 MQ Authorizations

MQ configuration requires UNIX user account and user group (principle/group) authorizations.

Table 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 alternate user while executing any messaging operation	`inq` on queue
OS login of alternate user while executing the messaging read operation	`get` on queue
OS login of alternate user while executing the messaging browse operation	`browse` on queue
OS login of alternate user while executing the messaging send operation	`put` on queue
OS login of alternate user dynamic queue specified as the replyToQueue	`crt`, `dlt` on queue manager, and `get`, `inq` on Model Queue

> **i** Note
>
> When a message is sent to a remote queue, WebSphere MQ checks the user authentication on the transmit queue.

If you specify `alter_user=yes` in `msgsend`, the alternate user is the operating system login that is running SAP ASE. If you do not specify `alter_user`, the alternate user is the SAP ASE login that is performing the MQ operation.

2.18 Querying MQ Information

You can query SAP ASE for information about MQ objects on a specified queue manager by using the show option of the `sp_msgadmin` stored procedure.

Query for the following information:

- The name of the queue manager
- All queues and their queue types belonging to the queue manager
- All channels and their types belonging to the queue manager

To prepare WebSphere MQ to use sp_msgadmin 'show:

1. In WebSphere MQ, start the queue manager that you want to make inquiries on.

2. Ensure that an MQ listener is running for the queue manager.
3. Start the command server of the queue manager.
4. Ensure that you have a queue called SYSTEM.ADMIN.COMMAND.QUEUE in the queue manager.

3 SQL Reference

This section describes global variables, stored procedures, functions, and syntax segments that you can use to manage and administer Active Messaging.

3.1 Transactional Messaging set Option

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

Syntax

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

Parameters

none
: Provides that messaging operations and database operations do not affect each other.

simple
: (Default) causes database operations to affect messaging operations, but messaging operations do not affect the database transaction.

full
: Provides full transactional behavior.

Examples

Example of none

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
```

Example of simple

In this example, `insert` is not aborted if `msgsend` fails:

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

Example of a rollback

In this example, msgsend is rolled back:

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

Example of full

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
```

Usage

- 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.

> **i** Note
>
> You cannot use `set transactional messaging` inside a transaction.

3.2 Message-Related Global Variables

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

These global variables are `char` datatypes, of length 16384. You can remove trailing blanks using `rtrim`.

3.2.1 @@msgcorrelation

Contains correlation from last message sent or read.

System	Description
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 correlation ID from the most recent message sent or received.

3.2.2 @@msgheader

Contains message header information from the most recent message received.

The format for `@@msgheader` is in XML. Functions that set `@@msgheader` include `msgrecv` and `msgconsume`.

The fields and descriptions for MQ are:

MQ 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: • 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.
Feedback	Feedback status.
FloatEncoding	This is the encoding for floating point numbers in the payload, and is a synthesized property derived from the Encoding field. If: • 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.

MQ Property Name	Description
Format	Format name of message data, can be an MQ-defined format name or an application-defined format name.
GroupID	Group identifier.
IntegerEncoding	Encoding for integers in the payload, and is a synthesized property that is derived from the Encoding field. If: • BigEndian – integers are big-endian. • LittleEndian – integers are little-endian. • Undefined – the endianness of integers is undefined.
LastMsgInGroup	If: • true – message is the last message of a group. • false – message is not the last message of a group.
MsgId	Message identifier.
MsgInGroup	If: • 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: • 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. • 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. If: • persistent – the message is a persistent message. • non-persistent – the message is a nonpersistent message.
PositiveActionNotification	This is a synthesized property derived from the Report field. The receiving application should generate a positive-action notification (PAN) report. If: • 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.

MQ Property Name	Description
PutApplName	This is the name of the application that puts the message in the queue.
PutApplType	This is the type of application that puts the message in the queue.
PutDate	This is the date when the message was put in the queue.
PutTime	This is the time when the message was put in the queue.
ReplyCorrelationId	A synthesized property, derived from the Report field. Denotes what to use as the correlation ID of the report message. • 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. • new – use a new message ID as the message ID of the report message. • original – use the message ID received 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.

The fields and descriptions for JMS are:

JMS 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: • persistent • non-persistent
priority	The message priority
redelivered	The redelivery status from the message

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

3.2.3 @@msgid

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

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`.

3.2.4 @@msgproperties

Contains message properties information from the most recent message received. This variable's format is in XML.

- (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 are inserted into `@@msgproperties`.
  - Since the name-value pairs in the MQRF header can have nonunique 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
MQPSTopic_2   */sports
```

Functions that set `@@msgproperties` include:

- (JMS) `msgrecv`, `msgconsume`
- (MQ) `msgrecv`

The value pairs that are extracted from the RF header if they are present include:

- MQPSCommand
- MQPSCompCode
- MQPSCorrelId
- MQPSDelOpts
- MQPSErrorId
- MQPSErrorPos
- MQPSIntData
- MQPSParmId
- MQPSPubOpts
- MQPSPubTime
- MQPSQMgrName
- MQPSQName
- MQPSReason
- MQPSReasonText
- MQPSRegOpts
- MQPSSeqNum
- MQPSStreamName
- MQPSStringData
- MQPSSubIdentity
- MQPSSubName
- MQPSSubUserData
- MQPSSubUserData
- MQPSTopic
- 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, double quotes are replaced by a single quote.

3.2.5 @@msgreplyqmgr

(MQ only) Contains the `ReplyToQmgr` name of the last message read.

3.2.6 @@msgreplytoinfo

Contains the name (`<provider_url>`, `<queue_name>`, `<topic_name>`, `<user_name>`) of the topic or queue name used for both sending and replying messages directly. Can be a permanent or temporary destination.

Functions that set `@@msgreplytoinfo` include:

- (JMS) `msgconsume`, `msgpublish`, `msgrecv`, `msgsend`
- (MQ) `msgrecv`, `msgsend`

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>`.

MQ only – can contain the syntax for remote_qmgr; `@@msgreplytoinfo` shows request/reply messaging showing support for the cluster queue manager using `@@msgcorrelation`:

For example, one SAP ASE server connects to the MASTER_MSCAI queue manager, and sends a message to Q1, located on the SLAVE_MSCAI remote queue manager, with the `replyToQueue` property specified as MASTERQ. Once you send `msgsend`, its value becomes the value of `@@msgreplytoinfo`:

```
select msgsend('d','ibm_mq:CH1/tcp/host1(1105)?
qmgr=MASTER,remote_qmgr=SLAVE,queue=Q1,alter_user=yes',
    message property 'replyToQueue=MASTERQ')
go
select @@msgreplytoinfo
go
IBM_MQ:CH1/tcp/host1(1105)?qmgr=MASTER,queue=MASTERQ
```

The other SAP ASE server connects to the queue manager SLAVE, and receives the previously sent message from Q1. The `@@msgreplytoinfo` global variable then includes the syntax for `remote_qmg`, so that the reply queue in this case is the remote queue.

```
select msgrecv('ibm_mq:CH2/tcp/host2(4810)?
qmgr=SLAVE,queue=Q1,alter_user=yes', option 'timeout=100')
go
select @@msgreplytoinfo
go
ibm_mq:CH2/tcp/host2(4810)?qmgr=SLAVE,remote_qmgr=MASTER,queue=MASTERQ
```

> **i Note**
>
> When using a `@@msgreplytoinfo` that contains the syntax `remote_qmgr` to send a reply message, `msgrecv`, whether the reply message reaches the correct remote queue manager or not, depends on how you have configured your WebSphere MQ.

3.2.7 @@msgschema

(JMS only) Contains the schema of the message or a null value. Contains the value of the property `<ase_message_body_schema>`.

See the description of the schema option in `msgsend` and `msgpublish`. Functions that set `@@msgchema` include `msgsend` and `msgpublish`.

3.2.8 @@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 `@@msgstatus` include `msgsend`, `msgpublish`, `msgrecv`, and `msgconsume`.

3.2.9 @@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`, and `msgconsume`.
- (MQ) `msgsend` and `msgrecv`.

3.2.10 @@msgtimestamp

Contains the timestamp included in the message last sent.

Functions that set the variable are: `msgsend`, `msgpublish`.

(MQ only) The following example shows request/reply messaging using both `@@msgreplytoinfo` and `@@msgcorrelation`

Session 1 (Requester)	Session 2 (Receiver)

Session 1 sends the request message to Q100, and expects the reply message on Q200. It sets the correlation to 0x123456:
`select msgsend('sender message',`
`'ibm_mq:channel1/TCP/host1(5678)'`
`+ '?qmgr=QM1'`
`+ ',queue=Q100',`
`option 'msgType=request',`
`message property`
`'correlationId=0x123456'`
`+ 'replyToQueue=Q200')`	

	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:
	`select msgrecv(`
	`'ibm_mq:channel1/TCP/host1(5678)'`
	`+ '?qmgr=QM1'`

Session 1 (Requester)	Session 2 (Receiver)

	`+ ',queue=Q100')`
	`select msgsend('receiver reply',`
	`@@msgreplytoinfo,`
	`option 'msgType=reply'`
	`message property`
	`'correlationId='`
	`+ @@msgcorrelation)`

Session 1 reads the reply message from Q200, wanting only message with correlation 0x123456:
`select msgrecv(`
`'ibm_mq:channel1/TCP/host1(5678)'`
`+ '?qmgr=QM1'`
`+ ',queue=Q200'`
`option 'timeout=30ss',`
`+ 'correlationID=0x123456')`	

3.3 msgheader and msgproperties Documents

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

```
<!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>
```

Examples

msgheader or msgproperties

These examples show `msgheader` or `msgproperties` documents for two select statements:

```
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.

3.4 SAP ASE-Specific Messages for JMS

To help with debugging, monitoring, and so forth, predefined properties specific to SAP ASE are included in the properties portion of the JMS message. These properties typically handle messages that either originate from another SAP ASE 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`.

Property	Description
ASE_RTMS_CHARSET	Character set encoding of sent data. Always use this.
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 msgsend. If `ASE_MSGBODY_FORMAT` is xml, this property contains the XML schema describing the payload. This schema is not truncated, even if its value exceeds 16K. Always use this.
ASE_MSGBODY_FORMAT	The format of the message body: • `xml` • `string` – in server character set • `binary` • `unicode` – unichar in network order Always use this.
ASE_ORIGIN	Name of the originating SAP ASE. Present with `diagserver`.
ASE_RTMS_VERSION	Version of SAP ASE using Active Messaging. Always use this.
ASE_SPID	SPID that sent the message. Present with `diagserver`.
ASE_TIMESTAMP	The timestamp of SAP ASE showing the time the message was sent Present with `diagserver`.
ASE_VERSION	Version of SAP ASE that published the message. Always use this.

Property	Description
ASE_VERSIONSTRING	Version string of the SAP ASE. Provides information about platform, build type, and so on. Useful for debugging. Present with `diagserver`.

3.5 Keywords

There are keywords specific to ASE Active Messaging, and functions in which these keywords can be legally used.

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

3.6 Syntax for Topics

Websphere MQ publishes and subscribes to topic strings that follow specific formats.

- A topic is generally in the form "topic/subtopic," such as "sport/baseball."
- You can specify a wildcard, such as "*" or "?" within a topic.
- When specifying multiple topics, separate the topics with a colon. For example, "`topic1:topic2:topic3:`", and so on.
- If a topic contains spaces or commas, place the entire topic list 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. If the topic also contains embedded double quotes, escape the embedded double quotes by using 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 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 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 names by using "::"; any single, nonescaped 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')
```

3.7 Built-In Functions

SQL functions, including their option strings, allow you to administer Active Messaging.

Use these SQL functions to:

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

3.7.1 msgconsume

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> as specified in *option_string value*.

Table 3: option and option_string values for msgconsume

option values	option_string values	Default	Description
`timeout`	timespec between -1, 0 – (231– 1)	-1	By default, `msgconsume` 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.Values are in number of milliseconds. `timeout` uses the `timespec` option.
`requeue`	`<string>`	None	The name of a destination, `queue`, or topic on which to `requeue` messages that SAP ASE cannot process. If you do not specify requeue, and the message cannot be processed, you see an error message.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, SAP recommends that the subscription be subscribed:

```
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:

```
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.
- `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.
- Only messages of types `message`, `text`, or `bytes` are handled. If a message is encounter that cannot processed, 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 cannot be handled are placed on the queue specified. The specified endpoint must exist on the same messaging service provider as the endpoint used in `msgconsume`.
- An error message is issued if the messaging provider issues messages of types other than `message`, `text`, or `bytes`, and if `requeue` is not specified.
- If the subscription is not subscribed, SAP ASE subscribes it automatically while running `msgconsume`.
- 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. See *Message-Related Global Variables* for more information about <msgheader> and <msgproperties>.
  You can use msgpropvalue to extract the values of a specific property from XML documents <msgheader> and <msgproperties>, and other related functions.

Permissions

You must have messaging_role to run msgconsume.

3.7.2 msgpropcount

Extracts and returns the number of properties or attributes in <msg_doc> from a <msgheader> and <msgproperties> document.

Syntax

```
msgpropcount_call ::= msgpropcount([<msg_doc>])
    <msg_doc> ::= <basic_character_expression>
    <prop_name>::= <basic_character_expression>
```

Parameters

msgpropcount_call
: makes the request to use the msgpropcount function.

<msg_doc>
: is the <msgheader> or <msgproperties> XML document in the form of <basic_character_expression>. If you do not specify <msg_doc>, msgpropcount uses the current value of <@@msgproperties>.

prop_name
: is the property name from which you want to extract a value or type in the form of <basic_character_expression>.

Examples

Example 1

This example assumes 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>
```

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

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

3.7.3 msgproplist

Extracts and returns from a <msgheader> and <msgproperties> document a string in the format of an <option_string> with all of the property attributes of <msg_doc>.

Syntax

```
msgproplist_call::= msgproplist([ msg_doc] [returns varchar | text]))
    msg_doc ::= basic_character_expression
    prop_name::= basic_character_expression
```

Parameters

msgproplist_call

makes the request to use the `msgproplist` function.

msg_doc

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

prop_name

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

returns varchar | text

specifies the format of the returning message.

Examples

Example 1

This example assumes 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.

Either of these retrieves the list of properties belonging to a message:

```
select msgproplist
```

```
select msgproplist(@@msgproperties)
```

Usage

- If the result of the `msgproplist` call is more than 16K, the result value contains the word "TRUNCATED". If this happens, specify returns text so that the results are not truncated. You must use other `msgprop` 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))
  ```

3.7.4 msgpropname

Extracts and returns the property name from a <msgheader> and <msgproperties> 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 <msg_doc>.

Syntax

```
msgpropname_call ::= msgpropname(integer[ ,<msg_doc>]), )
    <msg_doc> ::= <basic_character_expression>
    <prop_name>::= <basic_character_expression>
```

Parameters

integer
: is the index of the value.

msgpropname_call
: makes the request to use the msgpropname function.

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

Assumes 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.

Example 2

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

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

3.7.5 msgproptype

Extracts and returns from a `msgheader` and `msgproperties` document the message provider's property type for the `<msg_doc>` property with a name that equals `<prop_name>`. The result is a null value if `<msg_doc>` does not have a property with a name is equal to `<prop_name>`.

Syntax

```
msgproptype_call ::= msgproptype(prop_name [ , msg_doc] )
    msg_doc ::= basic_character_expression
    prop_name::= basic_character_expression
```

Parameters

msgproptype_call

makes the request to use the `msgproptype` function.

<msg_doc>

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

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)
```

3.7.6 msgpropvalue

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 <@@msgprpoperties> 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

Retrieves the value of the eighth property:

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

Example 3

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

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

3.7.7 msgpublish

(JMS) 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>.

<properties_clause>

is either an <option_string> or one of the options listed in the following tables. The options described in Table 3-7 and Table 3-8 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.

<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 you specify a recognized header property in both the message property and the message header clauses, the one in the message header clause takes precedence.

You see an error if you specify unrecognized options in the option_clause.

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

Examples

Example 1

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'
  ```
- Options you can specify for `msgpublish` for JMS:

Table 4: Values for the msgpublish option_string parameter

Option	Values	Default	Comments
`schema`	no, yes, "user_schema"	no	Enter one of these values: ○ user_schema – 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 SAP ASE 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. The schema is included in the message as ASE_MSGBODY_SCHEMA property
`type`	text or bytes	text	The message type to send.

If you use a property not listed in *Values for the msgpublish option_string parameter*, it is set as a property in the message properties of the message sent. Options and values for the properties_clause parameter:

Table 5: Values for the msgpublish properties_clause parameter

Option	Values	Default	Disposition	Comments
`correlation`	`<string>`	none	header	Client applications set correlation IDs to link messages together. SAP ASE sets the correlation ID specified by the application.

Option	Values	Default	Disposition	Comments
`mode`	persistent, non-persistent	persistent	header	When you enter: • persistent – the message is backed by the JMS provider, using stable storage. If the messaging provider fails before the message can be consumed and the mode is set to persistent, it is likely that the message will be saved. • non-persistent and the messaging provider fails – you may lose a message before it reaches the desired destination.
`priority`	1 to 9	4	header	The behavior of priority is controlled by the underlying message bus. The values mentioned here apply to JMS. Priorities from 1 – 4 are normal; priorities from 5 – 9 are expedited.
`replyqueue`	A string containing a `<queue_name>`	none	header	If the value of `<queue_name>` or `<topic_name>` is: • syb_temp – SAP ASE creates a temporary destination and sends information related to the newly created temporary destination as a part of the

Option	Values	Default	Disposition	Comments
`replytopic`	A string containing a `<topic_name>`	none	header	header information. SAP ASE then updates `<@@msgreplytoinfo>` as the temporary destination. 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 – SAP ASE does not create a new destination, using instead the one specified by the user.
`ttl`	0 – (2^{63}-1)	0	header	ttl refers to time-to-live on the messaging bus. SAP ASE is not affected by this. 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. A value of 0 indicates that the message never expires. ttl uses the timespec option

Permissions

You must have messaging_role to run `msgpublish`.

3.7.8 msgrecv

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 `<end_point>` is 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 option_string>

is a value shown in *MQ option and option_string values for msgrecv* for MQ, and *JMS option and option_string values for msgrecv* for JMS.

> **i** Note
>
> Unrecognized <option_string> names result in an error.

<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>

is one of these valid SQL datatypes: 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:channel1/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:channel1/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:channel1/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 (AA – CC) are read in browse mode, and CC is removed. The browse cursor is then set back to the beginning, and three messages (AA – DD) are read in browse mode, and DD is removed. 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 are no longer on the queue.

```
-- Browse cursor at the beginning, this will return 'AA'
select msgrecv(
    'ibm_mq:channel1/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:channel1/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:channel1/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:channel1/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:channel1/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:channel1/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:channel1/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:channel1/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:channel1/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

SonicMQ JMS – receives a message from the queue Q1 from the specified end_point, using the timeout option:

```
select msgrecv
     ('sonicmq_jms:tcp://mysonic:7223?queue=Q1,user=sonic_usr,
      password=sonic_pwd',option 'timeout=1000')
```

Example 7

(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 8

(JMS) this msgrecv call consumes only messages from queue.sample when the message property "Name" is "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

- See <@@msgheader >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.
- SAP ASE handles only messages of types message, text, or bytes. If SAP ASE 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 SAP ASE 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 use msgpropvalue to extract the values of a specific property from a <msgheader> and <msgproperties> document. See msgpropvalue.
- The general format of <msgheader> and <msgproperties> is described in *Message-Related Global Variables*.

MQ and msgrecv

These statements 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:

```
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 selector '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 selector '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 as the sort order used by SAP ASE.

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`.

3.7.8.1 MQ Options for msgrecv

Available `<options>` and `<option_string>` values for `msgrecv` properties for MQ.

Option Name	option_string Values and Descriptions
allMsgsInGroup	This option is ignored unless you specify `groupId`. If you set the property to:

Option Name	option_string Values and Descriptions
	• `yes` – all logical messages of a group must be present on the queue before the first message of a group is returned. • `no` – (default) not all logical messages of a group are required to be present on the queue before returning the first message of a group.
allSegments	If you set the property to: • `yes` – all messages of a segmented message must be present on the queue before the first message segment is returned. • `no` – (default) not all messages of a segmented message are required to be present before returning the first message segment.
browse	If you set the property to: • `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. • `null` – (default) the message is read and removed from the queue. The position option controls which message is read. If you set `browse` to 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) If you also: • 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.

Option Name	option_string Values and Descriptions
	• Do not specify `timeout` – the `msgrecv` operation blocks until a message appears in the queue that matches the selection criteria.

bufferLength	bufferLength-sized buffer is used to read the message.
	• 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:
	  ○ Its default of 4MB, or
	  ○ A value that is much larger than the actual length of the messages.
	Set the `maxMsgLength` queue property to the minimum allowed for the application to 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.
	The `<option_string>` values for `bufferLength` are:
	• sizespec
	• 0, or 1 – value
	`bufferLength` defaults to either the:
	• 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.
	0 indicates to use the default. For pub/sub messages, bufferLength must include the length of the message topics, including the MQRF header.

closeAfterRecv	If you set the property to: • `no` – (default) the queue remains open after the current msgrecv operation. • `yes` – the queue closes after the current msgrecv operation, allowing the queue to be reopened with a different input mode on subsequent msgrecv calls.
completeMsg	If you set the property to: • `yes` – (default) segmented messages are returned as a single message. • `no` – if there are segmented messages, each segment is returned as a separate message.
correlationId	Correlation ID of message to read, used in select statements to select specific messages in your queue. If you set the property to: • `null` – (default) • `string` – 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.

Option Name	option_string Values and Descriptions
formatName	The name of the expected message format. Values are: • `null` – (default) • `string` – MQ limits this string to 8 bytes. If specified, and the name `formatName` field of the message does not match, the message is not read. See the `requeue` option for more information.
groupid	Group ID of message to read. This is a `select` option. Values are: • `null` – (default) • `string` – 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.
inputMode	The values for `inputMode` open the MQ queue in the following ways: • `browse` – opened for browsing only. The queue manager produces an error when you attempt a destructive read. • `Qdefault` – (default) 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 inshared or exclusive mode by another MQ handle. `inputMode` is valid only for `msgrecv`. For any endpoint, specify `inputMode` either: • On the first `msgrecv` operation, or • After you specify `closeAfterRecv`. Attempting to change the value of `inputMode` across calls may cause unexpected results.
msgId	Message ID of message to read. As a selection option, you can use `msgId` to select specific messages in your queue. • `null` – (default) • `string` – MQ defines this field as `BYTE array` that can support binary values. To enter a binary string as the `msgId`, use "0x..." as the value. Do not add quote marks around value, as that is interpreted as a quoted string.
offset	Offset of message to read. Values are integer between -1, and 0 – `maxint`.

Option Name	option_string Values and Descriptions
	If -1, the offset is not specified. As a `select` option, you can use offset to select specific messages in your queue.
ordering	If you set the property to: • `physical` – (default) the messages are read in the order in which they appear on the queue. • `logical` – the messages are read in logical order according to groupId, sequenceId, and offsets.
position	Controls which message is returned. Depending on the `inputMode` value you specify, there are one or two "read" positions: • "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 If you set the property to: • `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. The MQ queue manager applies the following before determining what message to return: • The default ordering of the queue (priority, first-in, first-out) • Any selection criteria specified (`messageId`, `correlationId`, `groupId`, `seqenceId`, or `offset`)
requeue	This must be a full URI of the endpoints. The read message is requeued to the queue specified if: • `msgrecv` reads a message when `formatName` is specified. • The read message has a different `formatName`. • `requeue` is not null. Values are: • `null` – (default) • `string` – MQ limits a requeue to 48 bytes. 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.
sequenceId	Sequence ID of message to read. Values are an integer between -1 (default) to 9,999,999

Option Name	option_string Values and Descriptions
	If -1, the sequence ID is not specified. As a selection option, you can use `sequenceId` to select specific messages in your queue.
truncationAllowed	You can truncate the message when: • The buffer used to read the message (`bufferLength`, or length of the returned datatype). • The buffer is smaller than the length of the message. Specify as: • `yes` – to allow truncation. • `no` – (default) to not allow truncation. The read fails when the value is no and message is truncated.
timeout	Specifies the timeout. Values are `timespec` between -1, 0 – (2^{32}–1). If: • -1 – (default) there is no timeout. • timeout is specified as an integer – the value is to be taken in milliseconds. See `timespec` for more information

3.7.8.2 JMS Options for msgrecv

Available options and `option_string values` for `msgrecv` properties for JMS.

Option Name	option_string Values and Description
requeue	The name of a destination, queue, or topic on which to requeue messages that SAP ASE cannot process. If you do not specify `requeue` and the message cannot be processed, you see an error message. The specified endpoint must be on the same messaging provider as `msgconsume` and `msgrecv`. Values are: • None – (default) • `string`
timeout	By default, `msgrecv` blocks the message until it reads the next message from the message bus. The values, in numbers of milliseconds, are: • timespec • -1, 0 - (231- 1) The default is -1. If `timeout` is not -1, `msgrecv` returns a null value when the timeout interval lapses without reading a message.

3.7.9 msgsend

Provides a SQL interface to send messages to different service endpoints 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, and can be binary, character, 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.

<option_string>

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

```
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
```

Table 6: option_string

Parameter	Description
<option_string>	String describing the option you want to specify
<simple_identifier>	String that identifies the value of an <option>
<quoted_string>	String formed using the normal SQL conventions for embedded quotation marks
<integer_literal>	Literal specified by normal SQL conventions
<float_literal>	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 an option for MQ or JMS. Property names are case sensitive.

(Tibco JMS only) Iif you use a property not listed in *JMS Properties for msgsend*, it is set as a property in the message properties of the message sent.

(MQ only) The values of <properties_clause> differ based on what you specify in the rhfCommand option:

- 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.
- 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.
- For deregisterSubscriber, the <message_body> argument to msgsend is ignored. If the msgType is request, the reply message is sent to replyToQmgr and replyToQueue.
- For publish, the 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.
- 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`.

- 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.

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 SAP ASE 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.

<header_clause>

allows users to specify only those header properties that are specified in *MQ Properties for msgsend* and *JMS Properties for msgsend* for Tibco JMS. If you enter an unrecognized header property, you see an error message. If you specify a recognized header property in both the `<message property>` and the `<message header>` clauses, the one in the message header clause takes precedence. If you specify any unrecognized names in the `<message header>` parameter, you see an error message.

Examples

Example 1 (SonicMQ JMS)

Sends the message "hello" to the specified endpoint:

```
select msgsend('hello',
```

```
    'sonicmq_jms:tcp://mysonic:7223?queue=testq,user=xyz')
```

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 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:channel1/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 have been 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)

Displays the `clustQBinding = default` option in `msgsend`, where behavior is determined by property "DEFBIND" of the queue. If the value is "`open`," the behavior is same as `clustQBinding=bind`; otherwise, the value is the same as `clustQBinding=nobind`:

```
select msgsend(
    "M", "ibm_mq:CH1/TCP/box1(5599)?qmgr=INVC,queue=Q1,alter_user=yes",
          option "clustQBinding=default")
```

Example 9 (MQ)

Sends a report message. The correlation ID, reply queue, and report message data header have been 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 10 (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:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'groupId=theGroup,sequenceId=1')
select msgsend('message 2',
    'ibm_mq:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'groupId=theGroup,sequenceId=2')
select msgsend('message 3',
    'ibm_mq:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'groupId=theGroup,sequenceId=3')
select msgsend('message 4',
    'ibm_mq:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'groupId=theGroup,sequenceId=4,lastMsgInGroup=yes')
    commit
```

Example 11 (MQ)

Sends a datagram message. Various confirmation reports are requested, and 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 12 (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 13 (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:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'ordering=logical,msgInGroup=yes')
select msgsend('second logical message of the group',
    'ibm_mq:channel1/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:channel1/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:channel1/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:channel1/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:channel1/TCP/host1(5678)?qmgr=QM,queue=QUEUE.COMMAND',
    message property 'ordering=logical,lastMsgInGroup=yes')
commit
```

Example 14 (MQ)

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 SAP ASE, even though there is no user ID called “joe” on host1.

```
select msgsend('Hello world',
    'ibm_mq:channel1/TCP/host1(5678)?qmgr=joeQM,queue=QUEUE1,alter_user=yes')
```

Example 15 (MQ)

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')
```

Example 16 (MQ)

Displays the `clustQBinding=bind` option in `msgsend`. The local "INVC" queue manager is a member of the Q1 cluster queue, and Q1 is cluster queue:

```
select msgsend(
    "M", "ibm_mq:CH1/TCP/box1(5599)?qmgr=INVC,queue=Q1,alter_user=yes",
          option "clustQBinding=bind")
```

When you initially run this select statement, the MQOPEN call chooses the cluster queue manager to receive the message. Subsequent statements issued during the same SQL session are automatically routed to the same queue manager.

Example 17 (MQ)

Displays the `clustQBinding = nobind` option in `msgsend`. The cluster queue manager that receives the message is chosen each time:

```
select msgsend(
    "M", "ibm_mq:CH1/TCP/box1(5599)?qmgr=INVC,queue=Q1,alter_user=yes",
          option "clustQBinding=nobind")
```

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 SAP ASE according to *SAP ASE-specific JMS Messages*.
- 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 the 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. See *Message-Related Global Variables*.
- Use single apostrophes ('), not double quotation marks (''), around quoted option or property values.

> **Note**
>
> `msgsend` allows messages to be sent to a topic, if you specify topic=topic_name as the destination. However, SAP recommends that you do not do this, as it may cause unexpected behavior.

- Unrecognized options or properties are ignored, but unrecognized option or property values are flagged as an error.
- 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.
- These 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, not 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-SAP ASE application, or `completeMsg=yes` for an SAP ASE 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`.

3.7.9.1 MQ Options for msgsend

Available option parameters for `msgsend` properties for MQ.

Option Name	Values
msgType	• `datagram` – (default) • `request` – if you use this value, you must also specify the `replyQueue` property. • `reply` • `report` – if you use this value, you must also specify the `reportDataHeader` and `feedback` properties.
rfhCommand	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. 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 MQPSCommand set to the following: • `deletePublication` – set to `DeletePub`. The endpoint is the endpoint to the publishing stream queue. • `deregisterPublisher` – set to `DeregPub`. • `deregisterSubscriber` – set to `DeleteSub`. • `publish` – set to `Publish`. The endpoint is the endpoint to the publishing stream queue. • `registerPublisher` – set to `RegPub`. • `registerSubscriber` – set to `RegSub`. • `requestUpdate` – set to `ReqUpdate` The default is null.
alter_user	Values are: • `yes` – 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. • `no` – • null – (default) 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. **i Note** If the machine running MQ is not also running SAP ASE, 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 SAP ASE.
clustQBinding	This option allows users to specify if they want to put messages in the same instance. If you do not send a message to the cluster queue, this option is ignored. When you specify:

Option Name	Values
	• `bind` – WebSphere MQ chooses both the message's destination and the queue manager hosting it when it first opens the message, determining all MQPUT calls to the destination decided when the MPOPEN call was made. • `nobind` – WebSphere MQ chooses a different destination for the message each time a request is made for MQ to put a message in the queue, with the desgination being chosen each time MQPUT is executed using the cluster queue handler obtained by the MPOPEN call. Where the message goes is based on load-balancing considerations (if this option is enabled) and queue manager availability. • `default` – (default) is the destination is driven by the binding property defined at the cluster queue definition level. This behavior also occurs when you are using a cluster system but do not specify the clustQBinding option.

3.7.9.1.1 rfhCommand Option Properties

The properties that are effective when `rhfCommand` is set.

3.7.9.1.1.1 deletePublications

MQ `msgsend` properties when `rfhCommand` is set to `deletePublications`.

Property	Description
local	• `no` – (default) globally retained publications are deleted from all brokers in the network. • `yes` – only the retained publications published locally at this broker are deleted.
streamName	Name of the publication stream for the specified topics. Values are: • `null` – (default) • `string` If not specified, the default is the stream queue to which this MQRFH command message is sent. MQ limits this string to 48 bytes.
topics	You must supply at least one topic for this required property. If omitted, generates an error. The value is `string`. Use the format detailed in "Syntax for topics". Retained messages matching this topic are deleted.

3.7.9.1.1.2 deregisterPublisher

MQ `msgsend` properties if `rfhCommand` is set to `deregisterPublisher`.

Property	Descriptions
correlationAsId	If you specify: • `no` – (default) `correlationId` is not used as part of the publisher's traditional identity. • `yes` – `correlationId` is used as part of the publisher's traditional identity. Specify `correlationId`, but not as 0x00. • `generate` – a system-generated `correlationId` is used as part of the publisher's traditional identity.
deregAll	Returns an error if you specify `topics`. Options are: • `no` – (default) no registered topics are deregistered. • `yes` – all topics registered for this publisher are deregistered, and the `topics` property is ignored.
queueName	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. Options are: • `null` – (default) if null, defaults to the `replyToQueue`. • `string`
qmgrName	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. Options are: • `null` – (default) if null, defaults to `replyToQmgr`. • `string`
streamName	• `null` – (default) assumes `SYSTEM.BROKER.DEFAULT.STREAM`. • `string` – if not null, this is the name of the publication stream. MQ limits this string to 48 bytes.
topics	These are the topics that this publisher deregisters. Options are: • `null` – (default) • `string` Use the format detailed in "Syntax for topics". Returns an error if: • The `deregAll` property is set to `yes`. • `topics` is not null.

3.7.9.1.1.3 deregisterSubscriber

MQ `msgsend` properties when `rfhCommand` is set to `deregisterSubscriber`.

Property	Descriptions
correlationAsId	If you specify: • `no` – (default) `correlationId` is not used as part of the publisher's traditional identity. • `yes` – `correlationId` is used as part of the publisher's traditional identity. Specify `correlationId`, but not as 0x00. • `generate` – a system-generated `correlationId` is used as part of the publisher's traditional identity.
deregAll	Returns an error if you specify `topics`. Options are: • `no` – (default) no subscriber topics are deregistered. • `yes` – all topics for this subscriber are deregistered, and the topics property is ignored.
queueName	This is the subscriber's queue name, used to establish the traditional identity of the subscriber. Specify it as the same value you specified when you registered the subscriber. Options are: • null – (default) if null, defaults to the `replyToQueue`. • `string`
qmgrName	This is the subscriber's queue manager name, used to establish the traditional identity of the subscriber. Specify it as the same value you specified when you registered the subscriber. Options are: • null – (default) if null, defaults to `replyToQmgr`. • `string`
streamName	MQ limits this string to 48 bytes. Options are: • null – (default) assumes `SYSTEM.BROKER.DEFAULT.STREAM`. • `string` – if not null, this is the name of the publication stream.
topics	These are the topics that this subscriber deregisters. Options are: • null – (default) • `string` Use the format detailed in "Syntax for topics". Returns an error if: • The `deregAll` property is set to `yes`. • `topics` is not null.

3.7.9.1.1.4 publish

MQ `msgsend` properties when `rfhCommand` is set to `publish`.

Property	Description
anon	If you specify: • `no` – (default) the identity of the publisher is divulged by the MQ pub/sub broker. • `yes` – the identity of the publisher is not divulged by the MQ pub/sub broker. Ignored if `noReg` is `yes`.
correlationAsId	If you specify: • `no` – (default) `correlationId` is not used as part of the publisher's traditional identity. • `yes` – `correlationId` is used as part of the publisher's traditional identity. Specify `correlationId`, but not as 0x00. • `generate` – a system-generated `correlationId` is used as part of the publisher's traditional identity.
directReq	If you specify: • `no` – (default) the MQ pub/sub broker sends this publication to all subscribers. • `yes` – the MQ pub/sub broker sends this publication only to subscribers that registered specifying local. Ignored if noReg is yes.
integerData	The value is a number between -1, 0–(2^{32} – 1) If: • Not -1 – this is optional publisher-defined information that is included in the publication's MQRF header. • -1 – (default) Although MQ pub/sub allows multiple integerData tags in the MQRF header, Active Messaging supports only one.
local	If you specify: • `no` – (default) the MQ pub/sub broker sends this publication to all subscribers. • `yes` – the MQ pub/sub broker sends this publication only to subscribers that registered specifying local. Ignored if `noReg` is `yes`.
noReq	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: • `no` – (default) the MQ pub/sub broker performs an implicit registration, using the values set by `anon`, `local`, and `directReq`. • `yes` – the MQ pub/sub broker does not perform an implicit registration. The `anon`, `local`, and `directReq` properties are ignored. If the publisher is already registered, and `anon`, `local`, or `directReq` is set to `yes`, the existing registration is altered according to those properties.

Property	Description
otherSubsOnly	If you specify: • `no` – (default) the MQ pub/sub broker does not send this publication to this publisher, even if the publisher has a subscription on this publication. • `yes` – the MQ pub/sub broker sends this publication to this publisher if the publisher has a subscription on this publication.
publishSequenceId	The value is a number between -1, 0–(2^{32} – 1). If: • 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. • -1 – (default) the sequence number is not set.
publishTimeStamp	If you specify: • null – (default) the publication timestamp is not set • not null (integer) – this is the publication timestamp in the form of YYYYMMDDHHMMSSth, using universal time. The format is not validated.
queueName	This is the queue used to determine the publisher's traditional identity. This is also where subscribers can send direct requests to this publisher. Values are: • null – (default) • `string` – MQ limits this string to 48 bytes.
qmgrName	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. Values are: • null – (default) • `string` – MQ limits this string to 48 bytes.
retainPub	This is the queue used to determine the publisher's traditional identity. This is also where subscribers can send direct requests to this publisher. Values are: • `no` – (default) the MQ pub/sub broker sends this publication to this publisher if the publisher has a subscription on this publication. • `yes` – the MQ pub/sub broker does not send this publication to this publisher, even if the publisher has a subscription on this publication.
stringData	This is the queue used to determine the publisher's traditional identity. This is also where subscribers can send direct requests to this publisher. Options are: • null – (default) • `string` – If not null – this is optional publisher-defined information that is included in the publication's MQRF header. Although MQ pub/sub allows multiple stringData tags in the MQRF header, ASE Active Messaging supports only one.
topics	These are the topics on which this publication has information. Options are:

Property	Description
	• null – (default) • `string` Use the format detailed in "Syntax for topics". Wildcards are not allowed. This is a required property, and generates an error if omitted.

3.7.9.1.1.5 registerPublisher

MQ `msgsend` properties when `rfhCommand` is set to `registerPublisher`.

Property	Description
anon	• `no` – (default) MQ pub/sub broker divulges the identity of the publisher. • `yes` – MQ pub/sub broker does not divulge the identity of the publisher.
correlationAsId	• `no` – (default) `correlationId` is not used as part of the publisher's traditional identity. • `yes` – `correlationId` is used as part of the publisher's traditional identity. You must specify `correlationId`, but not as 0x00. • `generate` – a system-generated `correlationId` is used as part of the publisher's traditional identity.
directReq	• `yes` – the publisher is willing to accept direct request for publication information from other applications. 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. • `no` – (default) the publisher is not willing to accept direct request for publication information from other applications.
local	• `no` – (default) the MQ pub/sub broker sends this publication to all subscribers. • `yes` – the MQ pub/sub broker sends this publication only to subscribers that registered specifying Local.
queueName	This is the publisher's queue name, used to establish the traditional identity of the publisher. This is also where subscribers can send direct requests to this publisher. Options are: • null – (default) • `string` – MQ limits this string to 48 bytes
qmgrName	This is the queue name, used to determine the publisher's traditional identity. This is also where subscribers can send direct requests to this publisher. Options are: • null – (default) • `string` – MQ limits this string to 48 bytes

Property	Description
streamName	• null – (default) assumes `SYSTEM.BROKER.DEFAULT.STREAM`. • `string` – if not null, this is the name of the publication stream. MQ limits this string to 48 bytes.
topics	These are the topics on which the publisher provides information. This is a required property, and generates an error if omitted. Options are: • none – (default) • `string` Use the format detailed in "Syntax for topics". Wildcards are not allowed.

3.7.9.1.1.6 registerSubscriber

MQ `msgsend` properties when `rfhCommand` is set to `registerSubscriber`.

Property	Description
anon	• `no` – (default) MQ pub/sub broker divulges the identity of the subscriber. • `yes` – MQ pub/sub broker does not divulge the identity of the subscriber.
correlationAsId	• no – (default) `correlationId` is not used as part of the publisher's traditional identity. • yes – `correlationId` is used as part of the publisher's traditional identity. You must specify `correlationId`, but not as 0x00. • generate – a system-generated `correlationId` is used as part of the publisher's traditional identity.
dupsOk	• no – (default) the publication is not specified in the RFH command. • yes – the broker is allowed to occasionally deliver a duplicate publication to the subscriber.
informIfRet	• no – (default) the publication is not specified in the RFH command. • yes – the broker informs the subscriber if the publication is retained, by setting the `MQPSPubsOptsIsRetainedPub` in the MQRF header of the message sent to the subscriber.
local	• no – (default) the subscription is not specified in the RFH command. • 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.
inclStreamName	• no – (default) the publication is not specified in the RFH command.

Property	Description
	• yes – the broker adds the publication stream name in the MQRF header to each message that is forwarded to the subscriber.
newPubsOnly	• no – (default) the publication is not specified in the RFH command. • yes – the broker sends this publication only to this subscriber, and retained publications that exist at registration time are not sent.
pubOnReqOnly	• no – (default) the publication is not specified in the RFH command. • yes – the broker sends only new publications to this subscriber. Retained publications that exist at registration time are not sent.
pubsPersistence	• `asQueue` – (default) the publication is placed on the subscriber queue with the default persistence of the subscriber queue. • `asPublication` – the publication is placed on the subscriber queue with the same persistence as the original publication. • `persistent` – the publication is placed on the subscriber queue as a persistent message. • `non-persistent` – the publication is placed on the subscriber queue as a nonpersistent message.
queueName	This is the queue used to determine the subscriber's traditional identity. Values are: • `null` – (default) • `string` – MQ limits this string to 48 bytes
qmgrName	This is the queue manager used to determine the subscriber's traditional identity. Values are: • `null` – (default) • `string` – MQ limits this string to 48 bytes
streamName	• `null` – (default) the subscription is identified by its traditional identity. • `string` – if not null, this is the name of the publication stream
topics	These are the topics on which the subscriber wants to receive publications. This is a required property, and generates an error if omitted. Values are: • none – (default) • `string` Use the format detailed in "Syntax for topics".

3.7.9.1.1.7 requestUpdate

MQ `msgsend` properties when `rfhCommand` is set to `requestUpdate`.

Property	Description
topics	These are the topics that the subscriber is requesting. This is a required property that generates an error if omitted. Options are: • none – (default) • `string` Use the format detailed in "Syntax for topics".
streamName	Options are: • null – (default) assumes `SYSTEM.BROKER.DEFAULT.STREAM`. • `string` – if not null, this is the name of the publication stream.
qmgrName	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. Options are: • null – (default) if null, defaults to `replyToQmgr`. • `string` – MQ limits this string to 48 bytes.
queueName	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. Options are: • `null` – (default) • `string` – MQ limits this string to 48 bytes.
correlationAsId	If you specify: • `no` – (default) `correlationId` is not used as part of the subscriber's traditional identity. • `yes` – `correlationId` is used as part of the subscriber's traditional identity. Specify `correlationId`, but not as 0x00. • `generate` – a system-generated `correlationId` is used as part of the subscriber's traditional identity.

3.7.9.2 MQ Properties for msgsend

Valid MQ message `property <property_option_clause>` types and values for `msgsend`

Property Option	Values
arrivalReport	Arrival of this message to the final destination should generate a confirm-on-arrival (COA) report. You must specify `replyToQueue`. If you specify:

Property Option	Values
	• `no` – (default) the COA report is not generated. • `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.
correlationId	Clients set correlation ID to link messages together. Values are: • null – (default) • `string` – MQ limits this string to 24 bytes. 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; do not use quotes around the value. If `rfhCommand` is not null: • If `correlationId` is not null, a new correlation ID is not requested. If `correlationAsId` is `yes`, and `correlationId` is null, this is a separate traditional identity (one where correlation ID is empty). • For `rfhCommands` of `deletePublication`, `deregisterPublisher`, `publish`, and `registerPublisher`, the correlation ID specified is as part of the publisher's traditional identity.
deliveryReport	Delivery of this message from the final destination generates a confirm-on-arrival (COA) report. You must specify `replyToQueue`. If: • `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` – (default) the COA report is not generated.
exceptionReport	Expiration of this message or failure of this send generates an exception report. You must specify `replyToQueue`. If: • `no` – (default) the exception report is not generated. • `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.
expirationReport	The failure of this send generates an exception report. You must specify replyToQueue. If:

Property Option	Values
	• `no` – (default) the exception report is not generated. • `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.
expiry	The message's time-to-live on the queue manager. The values are a `timespec` between -1 and 214748364799. If the `timespec` is an integer, units are in milliseconds. Values are: • 0 – message does not expire. • -1 – (default) uses the default defined for the queue. **i Note** `expiry` is in tenths of a second, so this number is rounded to the tenths of a second before being passed to MQ.
feedback	For report messages, feedback is a code that indicates the nature of the report message. Values are an integer, and must range within MQFB_APPL_FIRST (65536) to MQFB_APPL_LAST (999999999). MQ defines one feedback code range each for: • System report messages • Application report messages
formatName	Application-defined property to pass information about the message formats. This property allows sending applications to set a format name that describes the message data. Values are: • null – (default) • `string` – MQ limits this string to 8 bytes. A receiving application can check `formatName` in `@@msgheader` to decide how to process the message data. Names beginning with "MQ" are reserved.
groupID	User-defined group. Values are: • null – (default) • `string` – MQ limits this string to 24 bytes. 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. If you do not specify `groupId` , but do specify one of the grouping properties, the queue manager generates the group name.

Property Option	Values
	Ignored if ordering is set to logical. All messages of a group must be sent in the same transaction.
lastMsgInGroup	Values are: • `no` – (default) • `yes` – marks a message as being the last logical message of a group. Use `yes` if you have a single logical message in a group by itself. You must send all messages of a group in the same transaction.
mode	Values are: • `default` – (default) the default defined for the queue is used. • `persistent` – the message is backed by the messaging provider, using stable storage. If the messaging provider fails before the message can be consumed, the message is likely to be saved. • `non-persistent` – if you use this and the messaging provider fails, you may lose a message before it reaches the desired destination.
msgId	When specified, WebSphere MQ replaces any existing message ID with the value specified for `msgId`. Values are: • null – (default) • `string` – MQ limits this string to 24 bytes. MQ defines this field as `unsigned char`, which indicates that it can support binary values. To enter a binary string as the `msgId`, use "0x..." as the value. Do not use quotes around the value.
msgInGroup	Values are: • `no` – (default) • `yes` – if the value is yes, this message is a logical message of a message group. For messages in a group, set this property to `yes` for all logical messages of the group, except the last one, which should have `lastMsgInGroup` set to `yes`. You must send all messages of a group in the same transaction.
msgLastSegment	Values are: • `no` – (default) • `yes` – 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, set the `msgLastSegment` for the message to `yes`. When the value is `yes` and ordering is set to physical, also set the `offset` property. Send all messages in a group in the same transaction.

Property Option	Values
msgSegment	Values are: • `no` – (default) • `yes` – if the value is `yes`, this message is a segment of a segmented message. For messages that are part of a single segment, set this property to `yes` for all segments except the last one, which should have `msgLastSegment` set to `yes`. When the value is `yes` and ordering is set to `physical`, also set the `offset` property. Send all messages in a group in the same transaction.
negativeActionReport	You must specify replyToQueue. If: • `no` – (default) the NAN report is not generated. • `yes` – when the retrieving application reads this message and acts negatively on it, a negative-action (NAN) report is generated.
offset	Values are an integer between -1 (default), 0 – `maxint` When the message is a segment of a segmented message, set offset to the byte offset of the current message within the logical message. -1 indicates that the offset is not specified. `offset` is: • Ignored unless `msgSegment` or `msgLastSegment` are also specified. • Ignored by `msgpublish`. • Ignored if `ordering` is set to `logical`. Send all messages of a group in the same transaction.
onNoDelivery	If: • `deadLetter` – (default) if the message cannot be delivered, it is placed in the dead-letter queue. • `discard` – the message is discarded by the queue manager.
ordering	When this property is: • `physical` – (default) 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 gaps 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	You must specify `replyToQueue`. If: • `no` – (default) the PAN report is not generated. • `yes` – when the retrieving application reads this message and acts positively on it, a positive-action notification (PAN) report is generated.

Property Option	Values
priority	Controls the priority of the message. Values are an integer: • -1 (default) • 0 to queue manager • `configured max priority` If: • -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	If: • `msgId` – (default) 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	If: • `new` – (default) the generated report message contains a new message ID. • `original` – the report message uses the same message ID as the message received.
replyToInputMode	The mode that the `replyToQueue` is opening. When you specify `replyToQueue`, the queue is automatically opened for subsequent input. This mode specifies the input mode that the replyToQueue is opening. This property is ignored if you do not specify `replyToQueue`. The modes have the following meanings: • `browse` – the queue is opened for browsing only. If you attempt to perform a destructive read, the queue manager issues an error message. • `Qdefault` – (default) the queue is opened in the default input mode as defined for the queue. • `shared` – the queue is opened in shared input mode. If the queue is already opened in exclusive mode by another MQ handle, you see an error message. • `exclusive` – the queue is opened in exclusive input mode. An error appears 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. If the queue is already opened in exclusive mode by another MQ handle, you see an error message.

Property Option	Values
	• `browse+exclusive` – the queue is opened for browsing, as well as for exclusive input mode. An error appears if the queue is already opened in shared or exclusive mode by another MQ handle.
replyToModel	The name of the model queue from which the reply queue is created, when the replyToQueue is a dynamic queue. Values are: • null – (default) • `string` – MQ limits this string to 48 bytes. If you do not specify `replyToQueue`, this property is ignored.
replyToQueue	The queue where the application expects a reply to a request message. Values are: • null – (default) • `string` The message type sent does not have to be `request`, as MQ does not enforce this. If the queue name specified ends with a "`*`", a system-generated dynamic queue name is generated with the specified prefix. If `replyToModel` and a dynamic queue name are specified, the dynamic queue is created from the model queue specified for `replyToModel`. You can obtain system-generated dynamic queue names after the send operation via the `@@msgreplytoinfo` session variable. When you specify a dynamic queue name, the current SAP ASE login must have "crt" authorization in the queue manager to create the dynamic queue. When a dynamic queue name is specified, manually delete the dynamic queue that is created if the receiving application does not do so. When `rfhCommand` is not null, specify `replyToQueue` to get responses from the MQ pub/sub broker.
replyToQmgr	Reserved for the queue manager where `replyToQueue` resides in the future. Currently, `replyToQueue` is always on the connected queue manager. Values are: • null – (default) • `string`
rfhCommand	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. If `rfhCommand` is null (the default), the message does not include the MQRF header. The message includes the MQRF header with any other value for `rfhCommand`, with the MQPSCommand set to the following: • `deletePublication` – set to `DeletePub`. The endpoint is the endpoint to the publishing stream queue.

Property Option	Values
	• `deregisterPublisher` – set to `DeregPub`. • `deregisterSubscriber` – set to `DeleteSub`. • `publish` – set to `Publish`. The endpoint is the endpoint to the publishing stream queue. • `registerPublisher` – set to `RegPub`. • `registerSubscriber` – set to `RegSub`. • `requestUpdate` – set to `ReqUpdate`. The message is sent to the endpoint you specify. For these options, specify the endpoint to the publishing stream queue: • `publish` • `deletePublication` For these options, specify the endpoint to the MQ pub/sub broker control queue: • `deregisterPublisher` • `deregisterSubscriber` • `registerPublisher` • `registerSubscriber` • `requestUpdate`
sequenceId	Used to sequence logical messages that are part of a group. The value is an integer between -1 – 9,999,999. -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. Send all messages of a group in the same transaction.

3.7.9.3 JMS Options for msgsend

Available `option option_string` types and values for `msgsend` for JMS.

Option Name	Values
schema	Values are: • `"user_schema"` – is a user-supplied schema describing the message_body. • `no` – indicates that no schema is generated and sent out as part of the message.

Option Name	Values
	• `yes` – indicates that SAP ASE generates an XML schema for the message. `yes` is meaningful only in a `message_body` that uses the parameter `select_for_xml`. `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. The schema is included in the message as the ASE_MSGBODY_SCHEMA property.
type	The type of message to send. Values are: • `text` – (default) • `bytes`

3.7.9.4 JMS Properties for msgsend

Valid JMS message `property <properties_option_string>` types and values for `msgsend`

Property Option	Values
correlation	Client applications set correlation IDs to link messages together. The value is a string. Default is: • none • header SAP ASE sets the correlation ID the application specifies.
mode	If the mode is: • `persistent` – the message is backed by the JMS provider, using stable storage. If the messaging provider fails before the message is consumed, it is likely the message is saved. • `non-persistent` and the messaging provider fails – you may lose a message before it reaches the desired destination. Default is: • persistent • header
priority	The behavior of priority is controlled by the underlying message bus. The values, 1 – 9, apply to Tibco JMS. Default is: • 4 • header Priorities from 1 – 4 are normal; priorities from 5 – 9 are expedited.
replyqueue	The value is a string containing a `queue_name`. Default is:

Property Option	Values
	• none • header If the value of `queue_name` is: • `syb_temp` – SAP ASE creates a temporary destination and sends information related to the newly created temporary destination as a part of the header information. SAP ASE then updates `@@msgreplytoinfo` as the newly created temporary destination. The type of the temporary destination is a queue. • A desination that already exists – SAP ASE does not create a new destination, using instead the one specified by the user.
replytopic	The value is a string containing a `topic_name`. Default is: • none • header If the value of `topic_name` is: • `syb_temp` – SAP ASE creates a temporary destination and sends information related to the newly created temporary destination as a part of the header information. SAP ASE then updates `@@msgreplytoinfo` as the newly created temporary destination. The type of the temporary destination is a topic. • A desination that already exists – SAP ASE does not create a new destination, using instead the one specified by the user.
ttl	`ttl` refers to time-to-live on the messaging bus. SAP ASE is not affected by this. Values are 0 – (263– 1). Default is: • 0 • header 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.

3.7.10 msgsubscribe

JMS only – provides a SQL interface to subscribe a topic for the current SAP ASE session.

Syntax

```
<msg_subscribe>::= msgsubscribe
    (<subscription_name>)
```

```
        <subscription_name>::=<basic_character_expression>
```

Parameters

subscription_name

is the name of the subscription to which you are subscribing. A <basic_character_expression>.

Examples

Exapmle 1

Tells the JMS messaging provider to begin holding messages published to the topic registered as "subscription_1":

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

Usage

- Before you specify a subscription with msgsubscribe or msgunsubscribe, 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 for the current SAP ASE session 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.
- 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')
```

3.7.11 msgunsubscribe

JMS only – provides a SQL interface to unsubscribe a topic for the current SAP ASE session.

Syntax

```
<msg_unsubscribe>::=msgunsubscribe
    (<subscription_name> [with {remove | retain}])
        <subscription_name>::=<basic_character_expression>
```

Parameters

<subscription_name>

is the name of the subscription to which you are subscribing. A <basic_character_expression>.

with {remove | retain}

removes or retains the durable subscription from the JMS message provider.

Examples

Example 1

Tells the JMS messaging provider to stop holding messages published to the topic registered as "subscription_1":

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

Usage

- Before you specify a subscription with `msgsubscribe` or `msgunsubscribe`, 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'
```

- `msgunsubscribe` stops any current subscription for the current SAP ASE session to the endpoint and filter specified by `<subscription_name>`. It returns a 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 SAP SE 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`.
  When a user logs out of SAP ASE, all subscriptions in that session become unsubscribed. The effect is same as running `msgunsubscribe` using the `with retain` option.
  When you unsubscribe a durable subscription using `with remove`, the subscriber definition is removed from JMS message provider, causing all the messages held by the subscriber definition to be missed:

```
<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 7: SQL Sessions

Session 1	Session 2

`select msgunsubscribe ('subscription_1' WITH RETAIN)`
`selectmsgconsume ('subscription_1')`
`...`
`selectmsgconsume ('subscription_1')`
`select msgunsubscribe ('subscription_1' WITH RETAIN)`	

	`select msgsubscribe('subscription_1')`
	`select msgconsume('subscription_1')`
	`...`
	`select msgconsume('subscription_1')`
	`select msgunsubscribe('subscription_1' WITH RETAIN)`

- 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')
  ```

3.7.12 Function Arguments and Specifications

Built-in functions accept certain arguments and specifications.

3.7.12.1 endpoint

(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.
JMS endpoints are opaque to SAP ASE, 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>]/<tcp>/<hostname>(<port>)
                <channel> ::= <channel_name>[(<channel_security>)]
                    <channel_name> ::= <identifier>
                    <channel_security> ::= ssl:SSLCIPH=<channel_ciph>
                    <channel_ciph> ::= <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 uppecase or lowercase.

<channel_name>

is optional and is the name of the MQ server-connection channel. MQ limits the length of a channel name to 20 characters (bytes). If you do not define <channel_name>, Active Messaging uses the server-connection channel "SYSTEM.DEF.SRVCONN" to connect to the queue manager.

channel_security

is the security property of the channel. If you do not specify <channel_security>, SAP ASE communicates with WebSphere MQ without any security protocols. The valid value for <channel_security> is ssl.

channel_ciph

works with <channel_security>, and specifies the SSLCIPH property value of the server connection channel, and must be a valid CipherSpec value for a WebSphere MQ client. The valid values for channel_ciph are:

Table 8: Valid CipherSpec Names for channel_ciph

CipherSpec Name	Hash Algorithm	Encryption Algorithm	Encryption Bits
NULL_MD5 [1]	MD5	None	0
NULL_SHA [1]	SHA	None	0
RC4_MD5_EXPORT [1]	MD5	RC4	40
RC4_MD5_US [2]	MD5	RC4	128
RC4_SHA_US [2]	SHA	RC4	128
RC2_MD5_EXPORT [1]	MD5	RC2	40
DES_SHA_EXPORT [1]	SHA	DES	56
RC4_56_SHA_EX-PORT1024 [3, 4, 5]	SHA	RC4	56
DES_SHA_EX-PORT1024 [3, 4, 5, 6]	SHA	DES	56

CipherSpec Name	Hash Algorithm	Encryption Algorithm	Encryption Bits
TRIPLE_DES_SHA_US [4]	SHA	3DES	168
TLS_RSA_WITH_AES_128_CBC_SHA [7]	SHA	AES	128
TSL_RSA_WITH_AES_256_CBC_SHA [7]	SHA	AES	256
AES_SHA_US [8]	SHA	AES	128

1. On OS/400, available when either AC2 or AC3 is installed.
2. On OS/400, available only when AC3 is installed.
3. Not available for z/OS.
4. Not available for OS/400.
5. Specifies a 1024-bit handshake key size.
6. Not available for Windows.
7. Available only for AIX, HP-UX, and Linux for Intel platform.
8. Available only for OS/400, AC3.

tcp

is the transport protocol, and it can be uppercase or lowercase. Specify tcp to communicate with MQ through SSL.

<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.

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 remote MQ queue manager that contains the target queue definition. MQ limits the length of a queue manager name to 48 characters (bytes). When using:

- `msgsend` – if you omit this option, the local queue manager is used to locate the queue objects. Omit this option to benefit from workload balancing a cluster queue.
- `msgreceive` – SAP ASE ignores this option.

Unlike with JMS support, you cannot specify a user name and password with the endpoint. MQ checks the authority of the related OS login. See “MQ security”.

Examples

Example 1

Sends a message to the queue manager, where the communication is through the SSL-enabled CH1 channel, and the cipher suite is NULL_MD5:

```
select msgsend('e',
    'ibm_mq:CH1(ssl:sslciph=NULL_MD5)/tcp/linuxxml1:1105?qmgr=MASTER_QM1,
    queue=Q2')
```

Example 2

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 3

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 4

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')
```

3.7.12.2 option_string

Specifies the general syntax and processing for <option_string>. Individual options are described in the functions that reference them.

Syntax

```
<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
```

Parameters

<option_string>

is the string describing the option you want to specify.

<simple_identifier>

is the string that identifies the value of an <option>.

<quoted_string>

is the string formed using the normal SQL conventions for embedded quotation marks.

<integer_literal>

is the literal specified by normal SQL conventions.

<float_literal>

is the literal specified by normal SQL conventions.

true

is a Boolean literal.

false

is a Boolean literal.

null

is a null literal.

<byte_literal>

has the form 0xHH, where each H is a hexadecimal digit.

Usage

For <option_string> usage, see msgsend.

3.7.12.3 timespec

Message options and property values that accept a time interval using the timespec function accept the following syntax as a time specification for both MQ and JMS.

Syntax

```
'timeout=<timespec>'
    <timespec> ::= <integer_number> [ <timespec_units> ]
        <timespec_units> ::= { <dd> | <hh> | <mi> | <ss> | <ms> }
```

Parameters

<dd>

is days.

<hh>

is hours.

<mi>

is minutes.

<ss>

is seconds.

<ms>

is milliseconds.

<timespec_units>

is milliseconds. If you do not provide <timespec_units>, the default is milliseconds.

Examples

Example 1

Shows the time specification for 100 days:

```
-- timeout specified as 100 days
select msgrecv('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 msgrecv('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:

```
Shows the time specification for 1,024 milliseconds:
-- timeout specified as 1,024 milliseconds
select msgrecv(
    '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')
```

3.7.12.4 sizespec

(MQ only) Message options and property values that accept a size accept the following syntax as a size specification.

Syntax

```
sizespec ::= <integer_number> [ <sizespec_units> ]
    <sizespec_units> ::= { M | K }
```

Parameters

<integer_number>

is the size.

K or k

is kilobytes.

M or m

is megabytes.

<sizespec_units>

is the size specification in megabytes (M) or kilobytes (K), or bytes. If you do not provide <sizespec_units>, the default is bytes.

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')
```

3.8 Stored Procedures

Active Messaging uses the stored procedures `sp_configure 'enable real time messaging'`, `sp_engine`, and `sp_msgadmin`.

`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.

> **i** Note
>
> `sp_addexeclass` does not accept MQ Q engines for the `anyengine` and `lastonline` parameters.

3.8.1 sp_configure 'enable real time messaging'

Enables or disables real-time messaging, or displays the current messaging configuration.

Syntax

```
sp_configure "enable real time messaging"
    [, <enable_or_disable>]
    [, <rtm_provider> | drop instance]
    [, <instance_name>]
```

Parameters

<enable_or_disable>

specifies whether or not to enable or disable the "real time messaging" option. Valid values are:

- 1 – enables real-time messaging.
- 0 – disables real-time messaging

If omitted, the current "`real time messaging`" configuration is returned

<rtm_provider>

specifies the type of active messaging provider you are enabling or disabling. Use this parameter when specifying JVMs and cluster servers. Valid values are:

- `eas_jms` – enables or disables "`real time messaging`" for EAServer only.
- `ibm_mq` – enables or disables "`real time messaging`" for IBM MQ only.
- `sonicmq_jms` – enables or disables "`real time messaging`" for SonicMQ JMS only.
- `tibco_jms` – enables or disables "`real time messaging`" for Tibco JMS only.

drop instance

removes the messaging-related configuration option for one instance.

<instance_name>

is the name of the instance you specify when creating a cluster server environment. If you do not specify this option, the current real-time messaging configuration specifies the cluster-wide option.

Examples

Example 1

Enables real-time messaging for all providers :

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

Example 2

Disables real-time messaging for all providers :

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

Example 3

Enables real-time messaging for MQ only:

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

Example 4 (Cluster Edition)

Enables real-time messaging for all Active Messaging providers on all instances in the cluster:

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

Example 5 (Cluster Edition)

Disables the IBM MQ-only instance “ase1,” if the client is logged in to “ase1”:

```
sp_configure "enable real time messaging", 0, "ibm_mq", "ase1"
```

The value of the instance-specific configuration option generated for “ase1” is 26. The configuration value of “real time” is byte, with the different bit representing different real-time features:

- 0x1 – all are enabled.
- 0x2 – `tibco_jms` is enabled.
- 0x4 – `ibm_mq` is enabled.
- 0x8 – `eas_jms` is enabled.
- 0x10 – `sonicmq_jms` is enabled.

Example 6

Displays the instance-specific enable real time messaging configuration option status on the instance “ase1”. The value is 26:

```
sp_configure "enable real time messaging", null, null, ase1
```

Example 7

Drops the instance-specific enable real time messaging configuration option on the instance “ase1”. After you run this procedure, “ase1” begins to use the cluster-wide enable real time messaging configuration option, and the status becomes 1.

```
sp_configure "enable real time messaging", 0, "drop instance", "ase1"
```

Example 8

Displays the cluster-wide enable real time messaging configuration option status. Its value is 1:

```
sp_configure "enable real time messaging"
```

Usage

Using this stored procedure does not overwrite your previous setting. For example, if you enable `tibco_jms`, then run this stored procedure to enable MQ, both MQ and `tibco_jms` become enabled. Disabling tibco_jms does not affect MQ, which continues to be enabled.

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
- SAP licenses

The Cluster Edition allows you to configure multiple servers to run as a shared-disk cluster. Multiple machines connect to a shared set of disks and a high-speed private interconnection (for example, a gigabit Ethernet), allowing SAP ASE to scale using multiple physical and logical hosts. In the cluster system used in the following examples, clients connect to a shared-disk cluster named “mycluster,” which includes the “ase1,” “ase2,” “ase3,” and “ase4” instances running on machines “blade1,” “blade2,” “blade3,” and “blade4,” respectively. In these examples, a single instance resides on each node.

3.8.2 sp_engine

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" , [ <engine_id>]
```

Parameters

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 SAP ASE sessions that prevent the engine from being offline. You cannot use this parameter to specify a Q engine.

<engine_id>

is the ID of the engine.

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`.This parameter is required for `offline`, `q_offline`, `can_offline`, `q_can_offline`, `shutdown`, and `q_shutdown`.This parameter is not required for `online`, `q_online`.

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.

offline

brings an engine offline. You can also use `<engine_id>` to specify an engine to bring offline. You cannot use this parameter to specify a Q engine.

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 SAP ASE sessions that prevent the engine from being offline. You must use `<engine_id>` to specify whether a Q engine can be taken offline.

q_online

brings the next Q engine online.

q_shutdown

forces a Q 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 parameter 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 using 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 using this query:

```
select engine, status from sysengines where status = "online"
```

- 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.
- In a cluster environment, `sp_engine` works only for the engines of the local instance.

Permissions

You must be a system administrator to bring engines online or take them offline.

3.8.3 sp_msgadmin

Configures and administers messaging-related information.

Syntax

```
sp_msgadmin 'config', ['jvmlogging', <logging_level>
    | 'jvmpropertyfile', <filepath>
    | 'jvmlogfile', <filepath>
    | 'jvmmaxthreads', <thread_number>
    | 'jvmminthreads', <thread_number>
    | 'jvmthreadtimeout', <thread_timeout>
    | 'jvm' , <jvm_parameter>]
```

```
sp_msgadmin 'default', 'login', <provider_name>, <provider_login>,
        <provider_password>
```

```
sp_msgadmin 'help'
    [, 'list' | 'register' | 'default' | 'remove']
```

```
sp_msgadmin 'list',
    [| 'login'[, <provider_name>, [<login_name>]
    | 'provider' [, <provider_name>]
```

```
    | 'subscription' [, <subscription_name>]]
```

```
sp_msgadmin 'register',
    ['provider', <provider_name>, <provider_class>,
        <messaging_provider_URL>
    | 'login', <provider_name>, <local_login>, <provider_login>,
        <provider_password> [, <role_name>]
    | 'subscription', <subscription_name>, endpoint[, selector
        [, <delivery_option> [, <durable_name>, <client_id>]]]]
```

```
sp_msgadmin 'remove',
    ['<provider>', <provider_name>
    | 'login', <provider_name>, <local_login> [, <role>]
    | 'subscription', <subscription_name>
```

```
sp_msgadmin 'show',
    <showtype>, <provider>[, <options_clause>]
```

Parameters

sp_msgadmin 'config'

allows you to specify various configurations for either the Java Virtual Machine (JVM), or the key repository file path for SAP ASE for using MQ SSL. The configured values take effect after you reenable ASE Active Messaging. The options for sp_msgadmin 'config' are:

- `'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 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'` – is the default value for logging_level, and returns information-level log information.
  - `'error'` – returns only error log information.
- `'jvmpropertyfile', <filepath>` – specifies the property file that JVM uses for your configuration.
  `<filepath>` can be any valid path for your property file, including the use of environment variables. The default value for the property file 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 the JVM log file filepath is `$SYBASE/$SYBASE_ASE/rtms.log`.
When you start a Java Active Messaging server in a cluster environment, the actual log file is a combination of the value and `@@nodename`. For example, if you run `sp_msgadmin` for node "s1," the actual JVM log file is `$SYBASE/$SYBASE_ASE/jrtms_s1.log`:

```
1> sp_msgadmin 'config', 'jvmlogfile', '$SYBASE/$SYBASE_ASE/
jrtms.log'
```

If the configured JVM log file:

- Has a file extension — such as `$SYBASE/$SYBASE_ASE/jrtms.log`, where the file name of `jrtms.log` includes the log file extension name — the real file name for instance "ase1" is `$SYBASE/$SYBASE_ASE/jrtms_ase1.log`.
- Does not have an extension file name — such as `$SYBASE/$SYBASE_ASE/jrtms`, where the file name is `jrtms` without a file extension — the real file name for instance "ase1" is `$SYBASE/$SYBASE_ASE/jrtms_ase1`.

- `'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. The value of `<thread_number>` must be greater than the value of `<jvmminthreads>`. The default value is 10.
- `'jvmminthreads', <thread_number>` – specifies the minimum 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 more, but must be fewer than the value of `<jvmmaxthreads>`. 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 the 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 Java to start with 500MB of allocated RAM. For more information on the Java -Xmx flag, see the Java Web site.
- `'ibmmq_keystore', <keystore_name>` – configures the key repository file path for SAP ASE to be able to send and receive messages to or from WebSphere MQ through SSL.
- `<keystore_name>` is the location of the key database file in which keys and certificates are stored.

sp_msgadmin 'default'

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

- `'login'` – when used with `'default'` specifies a default login.

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

- <provider_name> – is the messaging provider you are registering, which can be as many as 30 characters in length.
- <provider_login> – is the login name of the messaging provider that local_login maps to when connecting to the message provider. <provider_login> is also the default login the provider uses when sending or receiving messages from the <provider_login>.
- <provider_password> – is the password of the <provider_login>.

> **i Note**
>
> (Cluster environment only) If you use `sp_msgadmin default` to define the default login in a cluster environment, you can use the configuration over the entire cluster.

sp_msgadmin 'help'

provides syntax information about `sp_msgadmin` or its parameters.

sp_msgadmin 'list'

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

- `'login'[,`<provider_name>`,[`<login_name>`]` – lists information about a particular messaging provider login mapping, or about all messaging provider logins.
- `'provider'[,` <provider_name>`]` – specifies the message provider, and lists information about a particular messaging provider or about all message providers.
- `'subscription'[,` <subscription_name>`]` – lists information about a particular subscription or about all subscriptions.

sp_msgadmin 'register'

registers a messaging provider, login, or subscription. The options are:

- `sp_msgadmin 'register' provider` – registers the messaging provider, where:
  - <provider_name> – is the name of the messaging provider.
  - <provider_class> – is the class of the messaging provider you are adding. Valid values are:
    - `EAS_JMS`
    - `TIBCO_JMS`
    - `IBM_MQ`
    - `SONIC_MQ`
  - <messaging_provider_URL> – is the URL of the messaging provider you are registering.
- `sp_msgadmin 'register' 'login'` – registers a login mapping, where:
  - <provider_name> – is the name of a previously registered provider, and can be as many as 30 characters in length.
  - <local_login> – is an SAP ASE login that maps to the local login.
  - <provider_login> – is the login name of the messaging provider that local_login maps to when connecting to the message provider. <provider_password> – is the messaging provider password of the <provider_login>.

- `<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>`.

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

- `sp_admin 'register' 'subscription'` – registers a subscription, where:
  - `<subscription_name>` – is a subscription name.

> **i Note**
>
> You cannot use `sp_msgadmin 'register', 'subscription'` if `endpoint` is an MQ queue manager.

- `<selector>` – is a message filter that allows a client to select messages of interest. See `filters` in `msgrecv`.
- `<delivery_option>` – species 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 `<client_id>`.
- `<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, which species the subscription as durable. Otherwise, the subscription is nondurable.
  The `<client_id>` and `<durable_name>` combination identifies durable subscriptions with the message provider, and must be unique.
  `<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.

> **i Note**
>
> (Cluster environment only) If you use `sp_msgadmin 'register'` in a cluster environment to register provider, login, and subscription information the registration applies to the entire cluster.

sp_msgadmin 'remove'

removes a message provider, login, or subscription.

- `'provider', <provider_name>` – removes a messaging provider previously defined with:

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

  `<provider_name>` is an alias referring to the messaging provider you are removing.
- `'login',<provider_name>, <local_login> [, <role>]` – removes the mapping previously created between an SAP ASE login and a service provider login, defined by this call:

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

  Where:
  - `<local_login>` – is an SAP ASE login that maps to the local login.
  - `<role>` – is the role.
- `'subscription', <subscription_name>` – removes a subscription previously created by:

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

sp_msgadmin 'show'

displays the information about some MQ objects on a specified queue manager, where:

- `showtype` – allows you to specify the WebSphere MQ process or object to display:
  - `qmgr` – is the name of the queue manager.
  - `queues` – is all of the queues and their types that belong to the queue manager.
  - `channels` – is all the channels and their types that belong to the queue manager.
- `<provider>` – specifies the messaging provider. Use the full path format described in `endpoint`.
- `<option_string>` – is the list of options:
  - `timeout` – specifies the maximum time in milliseconds that the WebSphere MQ Administration Interface should wait for each reply message. Values are a `timespec` between 0 and (231–1), and the default is 30000 (30 seconds).
  - `replyqueue` – the command server returns the reply message to the queue. If you do not define the option, the command server returns the message to a dynamic queue, created by opening `SYSTEM.DEFAULT.MODEL.QUEUE`. Values are a `<string>`, and there is no default value.

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 10 minutes as the amount of time that a thread is idle before it is automatically destroyed:

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

Example 6 (JMS)

Starts the JVM with 500MB of RAM by using the -Xmx500m flag:

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

Example 7 (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 8 (JMS)

Specifies the default login that applies to all unmapped SAP ASE logins, when using a specified messaging provider for either sending or receiving:

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

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

Example 9 (JMS)

Specifies the default login:

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

Example 10 (JMS)

Lists the details for the user with a login of "loginsa":

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

Example 11 (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 12 (JMS)

Registers a login with the messaging provider login "jms_user1" and a specified password used for all SAP ASE logins that have sa_role permissions:

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

Example 13 (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 14 (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 15 (JMS)

Registers "subscription_1," a nondurable subscription.

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

> **i** Note
>
> You must first use `sp_msgadmin register, provider` to register "my_jms_provider".

Example 16 (JMS)

Removes the default login:

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

Example 17 (JMS)

Removes the SAP ASE login "ase_login1" associated with the messaging provider "my_jms_provider":

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

Example 18 (JMS)

Removes all logins for role sa_role on "my_jms_provider":

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

Example 19 (MQ)

Configures the key repository for SAP ASE to enable the use of SSL, where the key database file path is `/var/mqm/clients/ssl/KeyringClient.kdb`:

```
sp_msgadmin 'config', ibmmq_keystore,
    'var/mqm/clients/ssl/KeyringClient'
```

Example 20 (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 21 (MQ)

Displays the queue manager name from machine "bigcrunch" with a listening port of 3150:

```
sp_msgadmin 'show', 'QMGR', 'ibm_mq:/tcp/bigcrunch(3150)'
Name
--------------------------------------------------
TEST
```

Example 22

Displays the queue manager name. The queue manager is on machine "bigcrunch" with a listening port of 3150. The reply message is placed in the Q1 queue and the longest that SAP ASE waits for a reply message is 20 milliseconds:

```
sp_msgadmin 'show', 'QMGR', 'ibm_mq:channel1/tcp/bigcrunch(3150)',
    'timeout=20, replyqueue=Q1'
```

Example 23 (MQ)

Displays all of the queues on the queue manager. The reply message is placed in the Q1 queue and the longest that SAP ASE waits for a reply message is 20 milliseconds:

```
sp_msgadmin 'show', 'queues', 'ibm_mq:/tcp/bigcrunch(3150)',
    'timeout=20, replyqueue=Q1'
Name                                                            Type
----------------------------------------------------------------------
Q1                                                              LOCAL
SYSTEM.MQSC.REPLY.QUEUE                                         MODEL
RQ1                                                             REMOTE
AQ1                                                             ALIAS
...
```

Example 24 (MQ)

Displays all of the channels on the queue manager:

```
sp_msgadmin 'show', 'channels', 'ibm_mq:/tcp/bigcrunch(3150)'
Name                                                            Type
----------------------------------------------------------------------
SNCH1                                                           SENDER
SECH2                                                           SERVER
RCCH3                                                           RECEIVER
CHL5                                                            SRVCONN
...
```

Example 25 (SonicMQ)

Registers a subscription called “sub1” to the specified endpoint, and placed in the Q1 queue:

```
sp_msgadmin register, subscription, sub1,
    'sonicmq_jms:tcp://mysonic:7223??topic=T1,user=sonic_usr,
password=sonic_pwd'
```

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:
    1. Explicit login names and passwords, specified in the endpoint, if provided.
    2. Explicit login mapping for the current SAP ASE login.
    3. The default login name and password for the message provider, and the role corresponding to the SAP ASE login.
    4. The default login name and password for the message provider, with no specific role association.
    5. Null login name and password if none of the above apply.
  - You can modify the login mapping between the SAP ASE 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.
- `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, SAP recommends that you do not do this.
  - If you specify `<role_name>`, you must specify `<local_login>` as null.
- `sp_msgadmin 'config'`
  - `sp_msgadmin 'config'` is only available for JMS.
  - All `sp_msgadmin 'config'` parameters are stored in the sysattributes table. To retrieve the values, execute:

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

    See the Reference Manual: Tables for information about `sysattributes`.
  - All the parameters available for `sp_msgadmin 'config'` are dynamically configured except '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'`

4 Samples

Sample code illustrate the messaging functionality that is distributed with the Active Messaging option.

Samples	Description
Directories	The SAP 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 Active Messaging. • `jdbc` – JDBC samples using Active Messaging. 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 identical. For example, `queue_listener.bat` on a Windows platform may be `queue_listener` on a UNIX/Linux platform.
SQL code samples	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.
Java/JDBC code samples	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 SAP ASE as both a participant in messaging and as an application using the message bus.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

www.sap.com/contactsap

THE BEST RUN SAP