

## CHAPTER 10

# SQL Remote Administration

About this chapter This chapter describes general issues and principles for administering a running SQL Remote installation.

*🔗* For system-specific details, see the chapters "Administering SQL Remote for Adaptive Server Enterprise" on page 277 and "Administering SQL Remote for Adaptive Server Anywhere" on page 255.

### Contents

<b>Topic</b>	<b>Page</b>
Management overview	218
Managing SQL Remote permissions	219
Using message types	228
Running the Message Agent	239
Tuning Message Agent performance	243
Encoding and compressing messages	250
The message tracking system	252

## Management overview

This chapter describes administration issues for SQL Remote installations.

Administration of a deployed and running SQL Remote setup is carried out at a consolidated database.

- ◆ **Permissions** As a SQL Remote installation includes many different physical databases, a consistent scheme for users having permissions on remote and consolidated databases is necessary. A section of this chapter describes the considerations you need to make when assigning users permissions.
- ◆ **Configuring message systems** Each message system that is used in a SQL Remote installation has control parameters and other settings that must be set up. These settings are discussed in this chapter.
- ◆ **The Message Agent** The Message Agent is responsible for sending and receiving messages. While some details of how the Message Agent operates and the configuration options for it, are different for Adaptive Server Anywhere and Adaptive Server Enterprise, some concepts and methods are common to both. These common features are discussed here.
- ◆ **Message tracking** Administering a SQL Remote installation means managing large numbers of messages being handed back and forth among many databases. A section on the SQL Remote message tracking system is included to help you understand what the messages contain, when they are sent, how they are applied, and so on.
- ◆ **Log management** SQL Remote obtains the data to send from the transaction log. Consequently, proper management of the transaction log, and proper backup procedures, are essential for a smoothly running SQL Remote installation. While many details depend on the server you are running, the generic issues are discussed in this chapter.
- ◆ **Passthrough mode** This is a method for directly intervening at a remote site from a consolidated database. This method is discussed in this chapter.

## Managing SQL Remote permissions

Users of a database involved in SQL Remote replication are identified by one of the following sets of permissions:

- ◆ **PUBLISH** A single user ID in a database is identified as the publisher for that database. All outgoing SQL Remote messages, including both publication updates and receipt confirmations, are identified by the publisher user ID. Every database in a SQL Remote setup must have a single publisher user ID, as every database in a SQL Remote setup sends messages.
- ◆ **REMOTE** All recipients of messages from the current database, or senders of messages to the current database, who are immediately lower on the SQL Remote hierarchy than the current database must be granted REMOTE permissions.
- ◆ **CONSOLIDATE** At most one user ID may be granted CONSOLIDATE permissions in a database. CONSOLIDATE permissions identifies a database immediately above the current database in a SQL Remote setup. Each database can have only one consolidated database directly above it.

Information about these permissions are held in the SQL Remote system tables, and are independent of other database permissions.

### Granting and revoking PUBLISH permissions

When a database sends a message, a user ID representing that database is included with the message to identify its source to the recipient. This user ID is the **publisher** user ID of the database.

A publisher is required even for read-only remote databases within a replication system, as even these databases send confirmations to the consolidated database to maintain information about the status of the replication. The GRANT PUBLISH statement for remote Adaptive Server Anywhere databases is carried out automatically by the database extraction utility.

Granting and revoking PUBLISH permissions from Sybase Central

You can grant PUBLISH permissions from Sybase Central. You must connect to the database as a user with full system or database administrator permissions.

#### ❖ To set the publisher for a database:

- 1 Open the SQL Remote folder.

- 2 Double click Set Publisher, and select a user from the list.
- 3 Click OK to set the selected user as the database publisher.

You can also revoke PUBLISH permissions from Sybase Central.

❖ **To revoke PUBLISH permissions from Sybase Central:**

- 1 Open the SQL Remote folder. The current publisher is displayed in the right pane.
- 2 Right-click the current publisher.
- 3 Click Revoke Publish on the popup menu.

Granting and revoking PUBLISH permissions [Adaptive Server Anywhere]

For Adaptive Server Anywhere, PUBLISH permissions are granted using the GRANT PUBLISH statement:

```
GRANT PUBLISH TO userid ;
```

The *userid* is a user with CONNECT permissions on the current database. For example, the following statement grants PUBLISH permissions to user **S\_Beaulieu**:

```
GRANT PUBLISH TO S_Beaulieu
```

The REVOKE PUBLISH statement revokes the PUBLISH permissions from the current publisher:

```
REVOKE PUBLISH FROM userid
```

Granting and revoking PUBLISH permissions [Adaptive Server Enterprise]

For Adaptive Server Enterprise, PUBLISH permissions are granted using the **sp\_publisher** procedure:

```
sp_publisher userid
```

The *userid* is a user with CONNECT permissions on the current database. For example, the following statement grants PUBLISH permissions to user **S\_Beaulieu**:

```
exec sp_publisher 'S_Beaulieu'  
go
```

The database is set to have no publisher by executing the **sp\_publisher** procedure with no argument:

```
exec sp_publisher  
go
```

Notes on PUBLISH permissions

- ◆ Only one user ID on a database can hold PUBLISH permissions at one time. That user ID is called the **publisher** for the database.
- ◆ To see the **publisher** user ID in Sybase Central, open the SQL Remote folder; the **publisher** is shown on the right panel.

- ◆ To see the publisher user ID for an Adaptive Server Anywhere database outside Sybase Central, use the CURRENT PUBLISHER special constant. The following statement retrieves the **publisher** user ID:

```
SELECT CURRENT PUBLISHER
```

- ◆ To see the publisher user ID for an Adaptive Server Enterprise database outside Sybase Central, use the following statement:

```
SELECT name
FROM sysusers
WHERE uid = ( SELECT user_id
              FROM sr_publisher )
go
```

- ◆ If PUBLISH permissions is granted to a user ID with GROUP permissions, it is not inherited by members of the group.
- ◆ PUBLISH permissions carry no authority except to identify the publisher in outgoing messages.
- ◆ For messages sent from the current database to be received and processed by a recipient, the publisher user ID must have REMOTE or CONSOLIDATE permissions on the receiving database.
- ◆ The publisher user ID for a database cannot also have REMOTE or CONSOLIDATE permissions on that database. This would identify them as both the sender of outgoing messages and a recipient of such messages.
- ◆ Changing the user ID of a publisher at a remote database will cause serious problems for any subscriptions that database is involved in, including loss of information. You should not change a remote database publisher user ID unless you are prepared to resynchronize the remote user from scratch.
- ◆ Changing the user ID of a publisher at a consolidated database while a SQL Remote setup is operating will cause serious problems, including loss of information. You should not change the consolidated database publisher user ID unless you are prepared to close down the SQL Remote setup and resynchronize all remote users.

## Granting and revoking REMOTE and CONSOLIDATE permissions

REMOTE and CONSOLIDATE permissions are very similar. Each database receiving messages from the current database must have an associated user ID on the current database that is granted one of REMOTE or CONSOLIDATE permissions. This user ID represents the receiving database in the current database.

Setting REMOTE and CONSOLIDATE permissions

Databases directly below the current database on a SQL Remote hierarchy are granted REMOTE permissions, and the at most one database above the current database in the hierarchy is granted CONSOLIDATE permissions.

For Adaptive Server Anywhere, the GRANT REMOTE and GRANT CONSOLIDATE statements identify the message system and address to which replication messages must be sent.

For Adaptive Server Enterprise, the `sp_grant_remote` procedure sets REMOTE permissions, and the `sp_grant_consolidate` procedure sets CONSOLIDATE permissions.

CONSOLIDATE permissions must be granted even from read-only remote databases to the consolidated database, as receipt confirmations are sent back from the remote databases to the consolidated database. The GRANT CONSOLIDATE statement at remote Adaptive Server Anywhere databases is executed automatically by the database extraction utility.

### Granting REMOTE permissions

Each remote database must be represented by a single user ID in the consolidated database. This user ID must be granted REMOTE permissions to identify their user ID and address as a subscriber to publications.

Granting REMOTE permissions accomplishes several tasks:

- ◆ It identifies a user ID as a remote user.
- ◆ It specifies a message type to use for exchanging messages with this user ID.
- ◆ It provides an address to where messages are to be sent.
- ◆ It indicates how often messages should be sent to the remote user.

Granting REMOTE permissions is also referred to as adding a remote user to the database.

Sybase Central example

You can add a remote user to a database using Sybase Central.

❖ **To add a new user to the database, as a remote user:**

- 1 Open the Remote Users folder in the SQL Remote folder.
- 2 Double-click Add Remote User, and follow the instructions in the wizard.

❖ **To make an existing user a remote user:**

- 1 Open the SQL Remote folder, so the Remote Users folder is displayed in the *left* pane.

- 2 Open the Users and Groups folder, so that the user you want to make a remote user is displayed in the *right* pane.
- 3 Drag the user icon in the right pane to the Remote Users folder in the SQL Remote folder on the left pane. The Make User a Remote User window is displayed.
- 4 Select the message type from the list, enter an address, choose the frequency of sending messages, and click OK to make the user a remote user.

#### Adaptive Server Anywhere example

The following statement grants remote permissions to user **S\_Beaulieu**, with the following options:

- ◆ Use an SMTP e-mail system
- ◆ Send messages to e-mail address **s\_beaulieu@acme.com**:
- ◆ Send message daily, at 10 pm.

```
GRANT REMOTE TO S_Beaulieu
TYPE smtp
ADDRESS 's_beaulieu@acme.com'
SEND AT '22:00'
```

#### Adaptive Server Enterprise example

The following statement grants remote permissions to user **S\_Beaulieu** with the following options:

- ◆ Use the file-sharing system to exchange messages.
- ◆ Place messages in the directory **beaulieu** under the address root directory.

The address root directory (for both Adaptive Server Anywhere and Adaptive Server Enterprise) is indicated by the **SQLREMOTE** environment variable, if it is set. Alternatively, it is indicated by the **Directory** setting in the **FILE** message control parameters (held in the registry or INI file).

- ◆ Send messages every twelve hours:

```
exec sp_grant_remote 'S_Beaulieu',
    'file',
    'beaulieu',
    'SEND EVERY',
    '12:00'
go
```

### Selecting a send frequency

There are three alternatives for the setting the frequency with which messages are sent. The three alternatives are:

- ◆ **SEND EVERY** A frequency can be specified in hours and minutes.  
When any user with SEND EVERY set is sent messages, all users with the same frequency are sent messages also. For example, all remote users who receive updates every twelve hours are sent updates at the same times, rather than being staggered. This reduces the number of times the Adaptive Server Anywhere transaction log or Adaptive Server Enterprise stable queue has to be processed. You should use as few unique frequencies as possible.  
  
SQL Remote is not intended for up-to-the-minute replication. Frequencies of less than ten minutes are not recommended.
- ◆ **SEND AT** A time of day, in hours and minutes.  
Updates are started daily at the specified time. It is more efficient to use as few distinct times as possible than to stagger the sending times. Also, choosing times when the database is not busy minimizes interference with other users.
- ◆ **Default setting (no SEND clause)** If any user has no SEND AT or SEND EVERY clause, the Message Agent sends messages every time it is run, and then stops: it runs in batch mode.

### Granting CONSOLIDATE permissions

In the remote database, the publish and subscribe user IDs are inverted compared to the consolidated database. The subscriber (remote user) in the consolidated database becomes the publisher in the remote database. The publisher of the consolidated database becomes a subscriber to publications from the remote database, and is granted CONSOLIDATE permissions.

At each remote database, the consolidated database must be granted CONSOLIDATE permissions. When you produce a remote database by running the database extraction utility, the GRANT CONSOLIDATE statement is executed automatically at the remote database.

Adaptive Server  
Anywhere example

The following Adaptive Server Anywhere statement grants CONSOLIDATE permissions to the **hq\_user** user ID, using the VIM e-mail system:

```
GRANT CONSOLIDATE TO hq_user
TYPE vim
ADDRESS 'hq_address'
```

There is no SEND clause in this statement, so the default is used and messages will be sent to the consolidated database every time the Message Agent is run.



Adaptive Server Enterprise example      The following Adaptive Server Enterprise statement grants CONSOLIDATE permissions to user **hq\_user**:

```
exec sp_grant_consolidate 'hq_user', address
go
```

### Revoking REMOTE and CONSOLIDATE permissions

A user can be removed from a SQL Remote installation by revoking their REMOTE permissions. Revoking remote user permissions also drops any subscriptions for that user.

Revoking permissions from Sybase Central

You can revoke REMOTE permissions on both Adaptive Server Enterprise and Adaptive Server Anywhere from Sybase Central.

#### ❖ To revoke REMOTE permissions from Sybase Central :

- 1 Open the SQL Remote folder.
- 2 Open the Remote Users folder.
- 3 Right-click the user whose permission you wish to revoke, and select Revoke Remote from the popup menu.

Revoking permissions in Adaptive Server Anywhere

REMOTE and CONSOLIDATE permissions can be revoked from a user using the REVOKE statement. The following statement revokes REMOTE permission from user **S\_Beaulieu**.

```
REVOKE REMOTE FROM S_Beaulieu
```

DBA authority is required to revoke REMOTE or CONSOLIDATE access.

Revoking permissions in Adaptive Server Enterprise

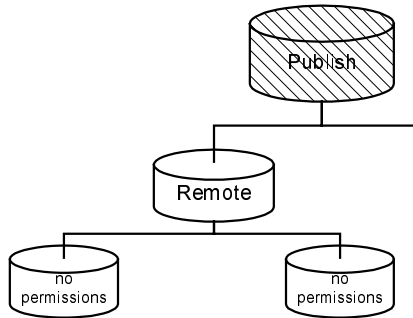
REMOTE permissions can be revoked from a user using the **sp\_revoke\_remote** procedure. This procedure takes a single argument, which is the user ID of the user. The following statement revokes REMOTE permission from user **S\_Beaulieu**.

```
exec sp_revoke_remote 'S_Beaulieu'
go
```

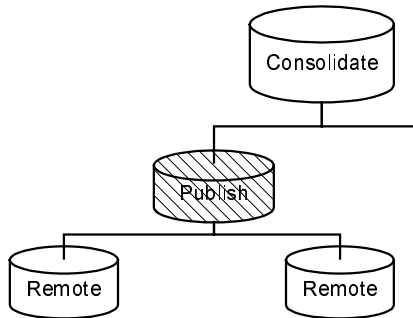
## Assigning permissions in multi-tier installations

Special considerations are needed for assigning permissions in multi-tier installations. The permissions in a three-level SQL Remote setup are summarized in the following diagrams. In each diagram one database is shaded; the diagram shows the permissions that need to be granted in that database for the user ID representing each of the other databases. The phrase "No permissions" means that the database is not granted any permissions in the shaded database.

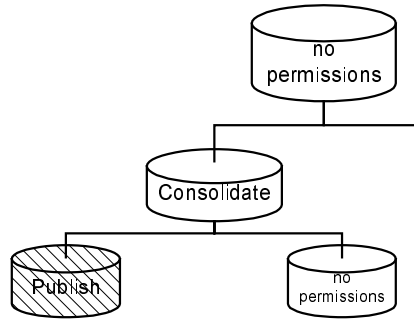
The following picture shows SQL Remote permissions, as granted at the consolidated site of a three-tier installation.



The following picture shows SQL Remote permissions, as granted at an internal site of a three-tier installation.



The following picture shows SQL Remote permissions, as granted at an internal site of a three-tier installation.



Granting the appropriate PUBLISH and CONSOLIDATE permissions at remote databases is done automatically by the database extraction utility.

## Using message types

SQL Remote supports several different systems for exchanging messages. The message systems supported by SQL Remote are:

- ◆ **file** Storage of message files in directories on a shared file system for reading by other databases.
- ◆ **ftp** Storage of message files in directories accessible by a file transfer protocol (ftp) link.
- ◆ **mapi** Microsoft's messaging API (MAPI) link, used in Microsoft Mail and other electronic mail systems.
- ◆ **smtp** Internet Simple Mail Transfer Protocol (SMTP/POP), used in Internet e-mail.
- ◆ **vim** Lotus's Vendor Independent Messaging (VIM), used in Lotus Notes and cc:Mail.

A database can exchange messages using one or more of the available message systems.

Operating system availability

Not all message systems are supported on all operating systems for which SQL Remote is available. The links are implemented as DLLs on Windows 3.x, Windows 95, and Windows NT.

The Message Agent for UNIX and NetWare operating systems supports the **file** link only, as a compiled library.

For more information

For full details on operating system availability, see the section on each message system:

- ◆ For more information on the **file** message system, see "The file sharing message system" on page 232.
- ◆ For more information on the **ftp** message system, see "The ftp message system" on page 233.
- ◆ For more information on the **smtp** message system, see "The SMTP message system" on page 233.
- ◆ For more information on the **mapi** message system, see "The MAPI message system" on page 235.
- ◆ For more information on the **vim** message system, see "The VIM message system" on page 237.

## Working with message types

Each message type definition includes the type name (**file**, **ftp**, **smtp**, **mapi**, or **vim**) and also the address of the publisher under that message type. The publisher address at a consolidated database is used by the database extraction utility as a return address when creating remote databases. It is also used by the Message Agent to identify where to look for incoming messages for the **file** system.

The address supplied with a message type definition is closely tied to the publisher ID of the database. Valid addresses are considered in following sections.

Before you can use a message system, you must set the publisher's address.

## Using Sybase Central to work with message types

You can create and alter message types in Sybase Central.

### ❖ To add a message type, using Sybase Central:

- 1 In the left pane, open the Message Types folder. The Message Types folder is inside the SQL Remote folder.
- 2 Double-click Add Message Type. The New Message Type window appears.
- 3 Enter one of the existing message type names, and a publisher address, in the appropriate fields. Click OK to save the definition in the database.

If you wish to change the publisher's address, you can do so by altering a message type.

### ❖ To alter a message type, using Sybase Central:

- 1 In the left pane, open the Message Types folder. The Message Types folder is inside the SQL Remote folder.
- 2 In the right pane, right-click the message type you wish to alter and select Properties from the popup menu. The Message Type Properties window appears.
- 3 Enter a new publisher address, in the appropriate fields. Click OK to save the definition in the database.

If you wish to drop a message type from the installation, you can do so.

❖ **To drop a message type, using Sybase Central:**

- 1 In the left pane, open the Message Type folder. The Message Type folder is inside the SQL Remote folder.
- 2 In the right pane, right-click the message type you wish to alter and select Delete from the popup menu.

**Using commands to work with message types**

❖ **To create a message type:**

- 1 Make sure you have decided on an address for the publisher under the message type.
- 2 Enter the command to create the message type.

For Adaptive Server Anywhere, use the CREATE REMOTE MESSAGE TYPE statement. This statement has the following syntax:

```
CREATE REMOTE MESSAGE TYPE type-name  
ADDRESS address-string
```

For Adaptive Server Enterprise, use the **sp\_remote\_type** procedure. This procedure takes the following arguments:

```
sp_remote_type type-name, address-string
```

In these statements, *type-name* is one of the message systems supported by SQL Remote, and *address-string* is the publisher's address under that message system.

If you wish to change the publisher's address, you can do so by altering the message type.

❖ **To alter a message type:**

- 1 Make sure you have decided on a new address for the publisher under the message type.
- 2 Enter the command to alter the message type.

For Adaptive Server Anywhere, use the ALTER REMOTE MESSAGE TYPE statement. This statement has the following syntax:

```
ALTER REMOTE MESSAGE TYPE type-name  
ADDRESS address-string
```

For Adaptive Server Enterprise, use the **sp\_remote\_type** procedure in the same way as creating a message type. This procedure takes the following arguments:

```
sp_remote_type type-name, address-string
```

In these statements, *type-name* is one of the message systems supported by SQL Remote, and *address-string* is the publisher's address under that message system.

You can also drop message types if they are no longer used in your installation. This has the effect of removing the publisher's address from the definition.

#### ❖ To drop a message type:

- ◆ Enter the command to drop the message type.

For Adaptive Server Anywhere, use the DROP REMOTE MESSAGE TYPE statement. This statement has the following syntax:

```
DROP REMOTE MESSAGE TYPE type-name
```

For Adaptive Server Enterprise, use the **sp\_drop\_remote\_type** procedure in the same way as creating a message type. This procedure takes the following arguments:

```
sp_drop_remote_type type-name
```

In these statements, *type-name* is one of the message systems supported by SQL Remote.

### Setting message type control parameters

Each message link has several parameters that govern aspects of its behavior. The following sections document these parameters.

Where the parameters are held

The message link control parameters are stored in the following places:

- ◆ **Windows 95 and Windows NT** In the registry, at the following location:

```
\\HKEY_CURRENT_USER
  \Software
    \Sybase
      \SQL Remote
```

- ◆ **Windows 3.x** In the file SQLANY.INI, in your SQL Remote installation directory.
- ◆ **NetWare** You should create a file named *dbremote.ini* in the *sys:\system* directory to hold the FILE system directory setting.
- ◆ **UNIX** The FILE system directory setting is held in the SQLREMOTE environment variable.

The *sqlremote* environment variable holds a path that can be used as an alternative to one of the control parameters for the file sharing system.

The parameters available for each message system are discussed in the following sections. Each section describes a single message system.

## The file sharing message system

	SQL Remote can be used even if you do not have a message system in place, by using the <b>file</b> message system.
Supported operating systems	The file sharing message system is supported on all platforms for which SQL Remote is available, for both Adaptive Server Enterprise and Adaptive Server Anywhere.
Addresses in the file message system	The <b>file</b> message system is a simple file-sharing system. A <b>file</b> address for a remote user is a subdirectory into which all their messages are written. To retrieve messages from their "inbox", an application reads the messages from the directory containing the user's files. Return messages are sent to the address (written to the directory) of the consolidated database.
Root directory for addresses	<p>The <b>file</b> system addresses are typically subdirectories of a shared directory that is available to all SQL Remote users, whether by modem or on a local area network. Each user should have a registry entry, initialization file entry, or SQLREMOTE environment variable pointing to the shared directory.</p> <p>You can also use the <b>file</b> system to put the messages in directories on the consolidated and remote machines. A simple file transfer mechanism can then be used to exchange the files periodically to effect replication.</p>

## FILE message control parameters

The FILE message system uses the following control parameters:

- ◆ **Directory** This is set to the directory under which the messages are stored. The setting is an alternative to the SQLREMOTE environment variable.
- ◆ **Debug** This is set to either YES or NO, with the default being NO. When set to YES, all file system calls made by the FILE link are displayed.

The FILE section of the *sqlany.ini* file (Windows 3.x) has the following entries:

```
[ FILE ]
Directory=path
Debug={ yes | no }
```

On NetWare, you should create a file named *dbremote.ini* in the *sys:\system* directory to hold the directory setting.



## The ftp message system

Supported operating systems	The ftp message system is supported on the Windows NT, Windows 95, and Windows 3.x operating systems.
Addresses for ftp	In the ftp message system, messages are stored in directories under a root directory on an ftp host. The ftp host and the root directory are specified by message system control parameters held in the registry or initialization file, and the address of each user is the subdirectory where their messages are held.

### FTP message control parameters

The ftp message system uses the following control parameters:

- ◆ **host** The host name of the computer where the messages are stored. This can be a host name (such as **ftp.powersoft.com**) or an IP address (such as 192.138.151.66).
- ◆ **user** The user name for accessing the ftp host.
- ◆ **password** The password for accessing the ftp host.
- ◆ **root\_directory** The root directory within the ftp host site, under which the messages are stored.
- ◆ **port** Usually not required. This is the IP port number used for the Ftp connection.
- ◆ **debug** This is set to either YES or NO, with the default being NO. When set to YES, debugging output is displayed.

## The SMTP message system

The Simple Mail Transfer Protocol (SMTP) is used in Internet e-mail products.

With the SMTP system, SQL Remote sends messages using Internet mail. The messages are encoded to a text format and sent in an e-mail message to the target database. The messages are sent using an SMTP server, and retrieved from a POP server: this is the way that many e-mail programs send and receive messages.

Supported operating systems	The SMTP message system is supported on the following operating systems: <ul style="list-style-type: none"> <li>◆ Windows 95</li> <li>◆ Windows NT</li> <li>◆ Windows 3.x</li> </ul>
-----------------------------	--

SMTP addresses  
and user IDs

To use SQL Remote and an SMTP message system, each database participating in the setup requires a SMTP address, and a POP3 user ID and password. These are distinct identifiers: the SMTP address is the destination of each message, and the POP3 user ID and password are the name and password entered by a user when they connect to their mail box.

**Separate e-mail account recommended**

It is recommended that a separate POP e-mail account be used for SQL Remote messages.

**Sharing SMTP/POP addresses**

The database should have its own e-mail account for SQL Remote messages, separate from personal e-mail messages intended for reading. This is because many e-mail readers will collect e-mail in the following manner:

- 1 Connect to the POP Host and download all messages.
- 2 Delete all messages from POP Host
- 3 Disconnect from POP Host.
- 4 Read mail from the local file or from memory

This causes a problem, as the e-mail program downloads and deletes all of the SQL Remote e-mail messages as well as personal messages. If you are certain that your e-mail program will not delete unread messages from the POP Host then you may share an e-mail address with the database as long as you take care not to delete or alter the database messages.

These messages are easy to recognize, as they are filled with lines of seemingly random text.

**SMTP message control parameters**

Before the Message Agent connects to the message system to send or receive messages, the user must either have a set of control parameters already set on their machine, or must fill in a window with the needed information. This information is needed only on the first connection. It is saved and used as the default entries on subsequent connects.

The SMTP message system uses the following control parameters:

- ◆ **local\_host** This is the name of the local computer. It is useful on machines where SQL Remote is unable to determine the local host name. The local host name is needed to initiate a session with any SMTP server. In most network environments, the local host name can be determined automatically and this entry is not needed.
- ◆ **TOP\_supported** SQL Remote uses a POP3 command called TOP when enumerating incoming messages. The TOP command may not be supported by all POP servers. Setting this entry to NO will use the RETR command, which is less efficient but will work with all POP servers. The default is YES.
- ◆ **smtp\_host** This is the name of the computer on which the SMTP server is running. It corresponds to the SMTP host field in the SMTP/POP3 login window.
- ◆ **pop3\_host** This is the name of the computer on which the POP host is running. It is commonly the same as the SMTP host. It corresponds to the POP3 host field in the SMTP/POP3 login window.
- ◆ **pop3\_userid** This is used to retrieve mail. The POP user ID corresponds to the user ID field in the SMTP/POP3 login window. You must obtain a user ID from your POP host administrator.
- ◆ **pop3\_password** This is used to retrieve mail. It corresponds to the password field in the SMTP/POP3 login window. If all of these five fields are set, the login window is not displayed.
- ◆ **Debug** When set to YES, displays all SMTP and POP3 commands and responses. This is useful for troubleshooting SMTP/POP support problems. Default is NO.

The SMTP section of the *sqlany.ini* file (Windows 3.x) has the following entries:

```
[SMTP]
smtp_host=smtp_host
pop3_host=pop3_host
pop3_userid=userid
pop3_password=password
Debug={ yes | no }
```

## The MAPI message system

The Message Application Programming Interface (MAPI) is used in several popular e-mail systems, such as Microsoft Mail and later versions of Lotus cc:Mail. SQL Remote supports the MAPI message system under Windows 3.x, Windows 95, and Windows NT.

Supported operating systems

The MAPI message system is supported on the following operating systems:

- ◆ Windows 95
- ◆ Windows NT
- ◆ Windows 3.x

MAPI addresses and user IDs

To use SQL Remote and a MAPI message system, each database participating in the setup requires a MAPI user ID and address. These are distinct identifiers: the MAPI address is the destination of each message, and the MAPI user ID is the name entered by a user when they connect to their mail box.

MAPI message and the e-mail inbox

Although SQL Remote messages may arrive in the same mail box as e-mail intended for reading, they do not in general show up in your e-mail inbox.

SQL Remote sends application-defined messages, which MAPI identifies and hides when the mailbox is opened. In this way, users can use the same e-mail address and same connection to receive their personal e-mail and their database updates, yet the SQL Remote messages do not interfere with the mail intended for reading.

If a message is routed via the Internet, the special message type information is lost. The message then does show up in the recipient's mailbox.

### MAPI message control parameters

The MAPI message system uses the following control parameters:

- ◆ **Debug** When set to YES, displays all MAPI calls and the return codes. This is useful for troubleshooting MAPI support problems. Default is NO.
- ◆ **Force\_Download** (default YES) controls if the MAPI\_FORCE\_DOWNLOAD flag is set when calling MapiLogon. This might be useful when using remote mail software that dials when this flag is set.
- ◆ **IPM\_Receive** This can be set to YES or NO (default NO). If set to YES, the MAPI link receives IPM messages, which are visible in the mailbox. If set to NO, the MAPI link receives IPC messages, which are not visible in the mailbox. This may be useful if your MAPI provider does not support IPC messages. Also, it may be useful when receiving messages over the Internet. In this case, the sender might not be using MAPI or the IPC attributes are have been lost.

- ◆ **IPM\_Send** This can be set to YES or NO (default NO). If set to YES, the MAPI link sends IPM messages, which are visible in the mailbox. If set to NO, the MAPI link sends IPC messages, which are not visible in the mailbox. This may be useful if your MAPI provider does not support IPC messages.
- ◆ **Profile** Use the specified Microsoft Exchange profile. You should use this if you are running the Message Agent as a service.

The MAPI section of the *sqlany.ini* file (Windows 3.x) has the following entries:

```
[ MAPI ]
IPM_Send={yes |no }
IPM_Receive={ yes |no }
Force_Download={yes |no }
Debug={yes | no}
```

## The VIM message system

The Vendor Independent Messaging system (VIM) is used in Lotus Notes and in some releases of Lotus cc:Mail.

Supported  
operating systems

The VIM message system is supported on the following operating systems:

- ◆ Windows 95
- ◆ Windows NT
- ◆ Windows 3.x

To use SQL Remote and a VIM message system, each database participating in the setup requires a VIM user ID and address. These are distinct identifiers: the VIM address is the destination of each message, and the VIM user ID is the name entered by a user when they connect to their mail box.

## VIM message control parameters

The VIM message system uses the following control parameters:

- ◆ **Path** This corresponds to the Path field in the cc:Mail login window. It is not applicable to and is ignored under Lotus Notes.
- ◆ **Userid** This corresponds to the User ID field in the cc:Mail login window.
- ◆ **Password** This corresponds to the Password field in the cc:Mail login window. If all of Path, Userid, and Password are set, the login window is not displayed.

- ◆ **Debug** When set to YES, displays all VIM calls and the return codes. This is useful for troubleshooting VIM support problems. Default is NO.
- ◆ **Receive\_All** When set to YES, the Message Agent checks all messages to see if they are SQL Remote messages. When set to NO (the default), the Message Agent looks only for messages of the application-defined type **SQLRemoteData**. This leads to improved performance in Notes.  
  
Setting **ReceiveAll** to YES is useful in setups where the message type is lost, reset, or never set. This includes setups including cc:Mail messages, or over the Internet.
- ◆ **Send\_VIM\_Mail** When set to YES, the Message Agent sends messages compatible with Adaptive Server Anywhere releases before 5.5.01, and compatible with cc:Mail. If this is set to YES, you should ensure that **Receive\_All** is set to YES also.

The VIM section of the *sqlany.ini* file (Windows 3.x) has the following entries:

```
[ VIM ]
Path=path
Userid=userid
Password=password
Debug={yes | no}
Receive_All = {yes | no}
Send_VIM_Mail = {yes | no}
```

## Running the Message Agent

The SQL Remote Message Agent is a key component in SQL Remote replication. The Message Agent handles both the sending and receiving of messages. It carries out the following functions:

- ◆ It processes incoming messages, and applies them in the proper order to the database.
- ◆ It scans the transaction log or stable queue at each publisher database, and translates the log entries into messages for subscribers.
- ◆ It parcels the log entries up into messages no larger than a fixed maximum size (50,000 bytes by default), and sends them to subscribers.
- ◆ It maintains the message tracking information in the system tables, and manages the guaranteed transmission mechanism.

### Executable names

The Message Agent for Adaptive Server Enterprise is named *ssremote.exe*, and the Message Agent for Adaptive Server Anywhere is named *dbremote.exe*.

☞ The Message Agent for Adaptive Server Enterprise uses a stable queue to hold transactions until they are no longer needed. For more information on the stable queue, see "How the Message Agent for Adaptive Server Enterprise works" on page 278.

## Message Agent batch and continuous modes

The Message Agent can be run in one of two modes:

- ◆ **Batch mode** In batch mode, the Message Agent starts, receives and sends all messages that can be received and sent, and then shuts down.

Batch mode is useful at occasionally-connected remote sites, where messages can only be exchanged with the consolidated database when the connection is made: for example, when the remote site dials up to the main network.

- ◆ **Continuous mode** In continuous mode, the Message Agent periodically sends messages, at times specified in the properties of each remote user. When it is not sending messages, it receives messages as they arrive.

Continuous mode is useful at consolidated sites, where messages may be coming in and going out at any time, to spread out the workload and to ensure prompt replication.

The options available depend on the send frequency options selected for the remote users. Sending frequency options are described in "Selecting a send frequency" on page 223.

❖ **To run the Message Agent in continuous mode:**

- 1 Ensure that every user has a sending frequency specified. The sending frequency is specified by a SEND AT or SEND EVERY option in the GRANT REMOTE statement (Adaptive Server Anywhere) or **sp\_grant\_remote** procedure (Adaptive Server Enterprise).
- 2 Start the Message Agent without using the `-b` command-line switch.

❖ **To run the Message Agent in batch mode:**

- ◆ Either:
  - ◆ Have at least one remote user who has neither a SEND AT nor a SEND EVERY option in their remote properties, or
  - ◆ Start the Message Agent using the `-b` command-line switch.

## Connections used by the Message Agent

The Message Agent uses a number of connections to the database server. These are:

- ◆ One global connection, alive all the time the Message Agent is running.
- ◆ One connection for scanning the log. This connection is alive during the scan phase only.
- ◆ One connection for executing commands from the log-scanning thread. This connection is alive during the scan phase only.
- ◆ One connection for the stable queue (Adaptive Server Enterprise only). This connection is alive during the scan and send phases.
- ◆ One connection for processing synchronize subscription requests. This connection is alive during the send phase only.
- ◆ One connection for each worker thread. These connections are alive during the receive phase only.



## Replication system recovery procedures

SQL Remote replication places new requirements on data recovery practices at consolidated database sites. Standard backup and recovery procedures enable recovery of data from system or media failure. In a replication installation, even if such recovery is achieved, the recovered database can be out of synch with remote databases. This can require a complete resynchronization of remote databases, which can be a formidable task if the installation involves large numbers of databases.

In short, recovery of the consolidated database from a failure at the consolidated site is only part of the task of recovering the entire replication installation.

Protection of the replication system against media failures has two aspects:

- ◆ **Backup and log management** Solid backup procedures and log management procedures for the consolidated database server are an essential part of recovery plans. Backup procedures protect against media failure on the database device. Using a transaction log mirror protects against media failure on the transaction log device.

☞ For more information about backup and log management procedures, see the sections "Transaction log and backup management" on page 262 and "Adaptive Server Enterprise transaction log and backup management" on page 286.

- ◆ **Message Agent configuration** The Message Agent command-line options provide ways for you to tune Message Agent behavior to match your backup and recovery requirements.

Message Agent configuration is discussed in the following pages.

### Replicating only backed-up transactions

By default, the Message Agent processes all committed transactions. When the Message Agent is run with the `-u` command-line switch, only transactions that have been backed up by the database backup commands are processed.

For Adaptive Server Anywhere, transaction log backup is carried out using Sybase Central or the `dbbackup` command-line utility, or off-line copying and renaming of the log file. For Adaptive Server Enterprise, transaction log backup is carried out using the **dump transaction** statement.

By sending only backed-up transactions, the replication installation is protected against media failure on the transaction log. Maintaining a mirrored transaction log also accomplishes this goal.


The `-u` switch provides additional protection against total site failure, if backups are carried out to another site.

## Ensuring consistent Message Agent settings

Some Message Agent settings need to be the same throughout an installation, and so should be set before deployment. This section lists the settings that need to be the same.

- ◆ **Maximum message length** The maximum message length for SQL Remote messages has a default value of 50K. This is configurable, using the Message Agent `-l` command-line switch. However, the maximum message length must be the same for each Message Agent in the installation, and may be restricted by operating system memory allocation limits.

Received messages that are longer than the limit are deleted as corrupt messages.

 For details of this setting, see "The Message Agent" on page 306.

## The Message Agent and replication security

Messages sent by the SQL Remote Message Agent have a very simple encryption that protects against casual snooping. However, the encryption scheme is not intended to provide full protection against determined efforts to decipher them.

## Tuning Message Agent performance

**Who needs to read this section?** If performance is not a problem at your site, you do not need to read this section.

There are several options you can use to tune the performance of the Message Agent. This section describes those options.

Sending messages and receiving messages are two separate processes. The major performance issues for these two processes are different.

- ◆ **Replication throughput** The major bottleneck for total throughput of SQL Remote sites is generally receiving messages from many remote databases and applying them to the database at the consolidated site. You can control this step by tuning the receive process of the Message Agent at the consolidated site.
- ◆ **Replication turnaround** The time lag from when data is entered at one site to when it appears at other sites is the turnaround time for replication. You can control this time lag.

### Tuning throughput by controlling Message Agent threading

It is assumed in this section that you are tuning the performance of a Message Agent that is running in continuous mode at a consolidated site.

Worker threads can be used by the Message Agent to apply incoming messages from remote users. This can improve throughput by allowing messages to be applied in parallel rather than serially.

Setting the number of worker threads

The number of worker threads is set on the Message Agent command line, using the `-w` switch. The following command line starts the Message Agent for Adaptive Server Enterprise with twenty worker threads applying messages:

```
ssremote -c "eng=..." -w 20
```

The default is to use no worker threads, so that all messages are applied serially. The maximum number of worker threads is 50.

Performance benefits from worker threads

For the Message Agent for Adaptive Server Anywhere, the performance advantage will be most significant when the server is on a system with a striped drive array.

For Adaptive Server Enterprise, the Message Agent will benefit even more if the Server is used with multiple engines configured.

What messages are applied in parallel

When worker threads are being used, messages from different remote users are applied in parallel. Messages from a single remote user are applied serially. For example, ten messages from a single remote user will be applied by a single worker thread in the correct order.

Deadlock is handled by re-applying the rolled back transaction at a later time.

Reading messages from the message system is single-threaded. Messages are read and the header information is examined (to determine the remote user and the correct order of application) before passing them off to worker threads to be applied.

Building messages and sending messages is single-threaded.

Open Client version

To use multiple worker threads with the Adaptive Server Enterprise Message Agent, you need to be using Open Client version 11.1 or above.

The Message Agent prints a message and then does not use worker threads when pre-11.1 versions are being used. The Open Client version is displayed in the first few lines of the Message Agent output.

## Tuning throughput by caching messages

The Message Agent caches incoming messages in a configurable area of memory as it reads them.

Specifying the message cache size

The size of the message cache is specified on the Message Agent command line, using the `-m` command-line switch.

The `-m` option specifies the maximum amount of memory to be used by the Message Agent for building messages. The allowed size can be specified as  $n$  (in bytes),  $nK$ , or  $nM$ . The default is 2048K (2M).

Example

The following command line starts an Adaptive Server Anywhere Message Agent using twelve Megabytes of memory as a message cache:

```
dbremote -c "eng=..." -m 12M
```

How messages are cached

When transactions are large, or messages arrive out of order, they are stored in memory by the Message Agent until the message is to be applied. This caching of messages prevents rereading of out-of-order messages from the message system, which may lower performance on large installations. It is especially important when messages are being read over a WAN (such as Remote Access Services or POP3 through a modem). It also avoids contention between worker threads reading messages (a single threaded task) because the message contents are cached.

When the memory usage specified using the `-m` switch is exceeded, messages are flushed in a least-recently-used fashion.

This switch is provided primarily for customers considering a single consolidated database for thousands of remote databases.

## Tuning incoming message polling

When running a Message Agent in continuous mode, typically at a consolidated database site, you can control how often it polls for incoming messages, and how "patient" it is in waiting for messages that arrive out of order before requesting that the message be resent. Tuning these aspects of the behavior can have a significant effect on performance in some circumstances.

### Issues to consider

The issues to consider when tuning the message-receiving process are similar to those when tuning the message-sending process.

- ◆ **Regular messages** Your choices dictate how often the Message Agent polls for incoming messages from remote databases.
- ◆ **Resend requests** You can control how many polls to wait until an out-of-order message arrives, before requesting that it be resent.
- ◆ **Processing incoming messages** If your polling period for incoming messages is too long, compared to the frequency with which messages are arriving, you could end up with messages sitting in the queue, waiting to be processed. If your polling period is too short, you will waste resources polling when no messages are in the queue.

☞ For more information on the message sending process, see "Tuning the message sending process" on page 247.

### Polling interval

By default, a Message Agent running in continuous mode polls one minute after finishing the previous poll, to see whether new messages have arrived. You can configure the polling interval using the `-rd` command-line option.

The default polling interval from the end of one poll to the start of another is one minute. You can poll more frequently using a value in seconds, as in the following command line:

```
dbremote -rd 30s
```

Alternatively, you can poll less frequently, as in the following command line, which polls every five minutes:

```
dbremote -rd 5
```

Setting a very small interval may have some detrimental impact on overall system throughput, for the following reasons:

- ◆ Each poll of the mail server (if you are using e-mail) places a load on your message system. Too-frequent polling may affect your message system and produce no benefits.
- ◆ If you do not modify the Message Agent patience before it assumes that an out of sequence message is lost, and requests it be sent again, you can flood your system with resend requests.

In general, you should not use a very small polling interval unless you have a specific reason for requiring a very quick response time for messages.

Setting larger intervals may provide a better overall throughput of messages in your system, at the cost of waiting somewhat longer for each message to be applied. In many SQL Remote installations, optimizing turnaround time is not the primary concern.

## Requesting resends

If, when the Message Agent polls for incoming messages, one message is missing from a sequence, the Message Agent does not immediately request that the message be resent. Instead, it has a default **patience** of one poll.

If the next message expected is number 6 and message 7 is found, the Message Agent takes no action until the next poll. Then, if no new message for that user is found, it issues a resend request.

You can change the number of polls for which the Message Agent waits before sending a request using the `-rp` command-line option. This option is often used in conjunction with the `-rd` option that sets the polling interval.

For example, if you have a very small polling interval, and a message system that does not preserve the order in which messages arrive, it may be very common for out-of-sync messages to arrive only after two or three polls have been completed. In such a case, you should instruct the Message Agent to be more patient before sending a resend request, by increasing the `-rp` value. If you do not do this, a large number of unnecessary resend requests may be sent.

## Example

Suppose there are two remote users, named **user1** and **user2**, and suppose the Message Agent command line is as follows:

```
dbremote -rd 30s -rp 3
```

In the following sequence of operations, messages are marked as *userX.n* so that **user1.5** is the fifth message from user1. The Message Agent expects messages to start at number 1 for both users.

At time 0 seconds:

- 1 The Message Agent reads user1.1, user2.4
- 2 The Message Agent applies user1.1
- 3 The Message Agent patience is now user1: N/A, user2: 3, as an out of sequence message has arrived from user 2.

At time 30 seconds:

- 1 The Message Agent reads: no new messages
- 2 The Message Agent applies: none
- 3 The Message Agent patience is now user1: N/A, user2: 2

At time 60 seconds:

- 1 The Message Agent reads: user1.3
- 2 The Message Agent applies: no new messages
- 3 The Message Agent patience: user1: 3, user2: 1

At time 90 seconds:

- 1 The Message Agent reads: user1.4
- 2 The Message Agent applies: none
- 3 The Message Agent patience user1: 3, user2: 0
- 4 The Message Agent issues resend to user2.

When a user receives a new message, it resets the Message Agent patience even if that message is not the one expected.

## Tuning the message sending process

The turnaround time for replication is governed by how often each sites sends messages and how often each site polls for incoming messages. To achieve a small time lag between data entry and data replication, you can set a small value for the `-sd` Message Agent command-line option, which controls the frequency for polling to see if more data needs to be sent.

### Issues to consider

The issues to consider when tuning the message-sending process are similar to those when tuning the incoming-message polling frequency:

- ◆ **Regular messages** Your choices dictate how often updates are sent to remote databases.
- ◆ **Resend requests** When a remote user requests that a message be resent, the Message Agent needs to take special action that can interrupt regular message sending. You can control the urgency with which these resend requests are processed.
- ◆ **Number and size of messages** If you send messages very frequently, there is more chance of small messages being sent. Sending messages less frequently allows more instructions to be grouped in a single message. If a large number of small messages is a concern for your message system, then you may have to avoid using very small polling periods.

☞ For more information on tuning polling for the incoming-messages, see "Tuning incoming message polling" on page 245.

## Polling interval

You control the interval to wait between polls for more data from the transaction log to send using the `-sd` command-line option, which has a default of one minute. The following example sets the polling interval to 30 seconds:

```
dbremote -sd 30s ...
```

Alternatively, you can poll less frequently, as in the following command line, which polls every five minutes:

```
dbremote -sd 5
```

Setting a very small interval may have some detrimental impact on overall system throughput, for the following reasons:

- ◆ Too-frequent polling produces many short messages. If the message load places a strain on your message system, throughput could be affected.

Setting larger intervals may provide a better overall throughput of messages in your system, at the cost of waiting somewhat longer for each message to be applied. In many SQL Remote installations, optimizing turnaround time is not the primary concern.



## Resending messages

When a user requests that a message be resent, the message has to be retrieved from early in the transaction log. Going back in the transaction log to retrieve this message and send it causes the Message Agent to interrupt the regular sending process. If you are tuning your SQL Remote installation for optimum performance, you must balance the urgency of sending requests for resent messages with the priority of processing regular messages.

The `-ru` command-line option controls the urgency of the resend requests. The value for the parameter is a time in minutes (or in other units if you add `s` or `h` to the end of the number), with a default of zero.

To help the Message Agent delay processing resend requests until more have arrived before interrupting the regular message sending activity, set this option to a longer time.

The following command line waits one hour until processing a resend request.

```
dbremote -ru 1h ...
```

## Encoding and compressing messages

As messages pass through e-mail and other message systems, there is a danger of them becoming corrupted. For example, some message systems use certain characters or character combinations as control characters.

Message size affects the efficiency with which messages pass through a system. Compressed messages can be processed more efficiently by a message system than uncompressed messages. On the other hand, carrying out compression can itself take a significant amount of time.

### SQL Remote encoding and compression

SQL Remote has a message encoding and compression scheme built in to the Message Agent. The scheme provides the following features:

- ◆ **Compatibility** The system can be set up to be compatible with previous versions of the software.
- ◆ **Compression** You can select a level of compression for your messages.
- ◆ **Encoding** SQL Remote encodes messages to ensure that they pass through message systems uncorrupted. The encoding scheme can be customized to provide extra features.

### Settings for compatibility

To be compatible with previous versions of the software, you should set the database option `COMPRESSION` to be -1 (minus one) at each database running the Version 6 software. This setting ensures that messages are sent out in a format compatible with older versions of the software.

If you upgrade the consolidated database Message Agent first, you should set its `COMPRESSION` database option to -1. As each remote site in your replication system is upgraded to Version 6, you can change its setting of the `COMPRESSION` option to a value between 0 (no compression) and 9 (maximum compression). This allows you to take advantage of compression features on messages being sent to the consolidated database. Once all remote sites are upgraded, you can set the consolidated site Message Agent `COMPRESSION` option to a value other than -1.

In addition, setting `COMPRESSION` to a value other than -1 allows you to take advantage of the Version 6 message encoding improvements.

## The encoding scheme

The default message-encoding behavior of SQL Remote is as follows:

- ◆ For message systems that can use binary message formats, no encoding is carried out.

- ◆ Some message systems, including SMTP, VIM, and MAPI, require text-based message formats. For these systems, an encoding DLL (*dbencod.dll* and *ssencod.dll*) translates messages into a text format before sending. The message format is unencoded at the receiving end using the same DLL.
- ◆ You can instruct SQL Remote to use a custom encoding scheme. The tools for building a custom encoding scheme are described in the following section.
- ◆ If the COMPRESSION database option is set to -1, then a Version 5 compatible encoding is carried out for all message systems.

## Creating custom encoding schemes

You can implement a custom encoding scheme by building a custom encoding DLL. You could use this DLL to apply special features required for a particular messages system, or to collect statistics, such as how many messages or how many bytes were sent to each user.

The header file *dbrmt.h*, installed into the *h* subdirectory of your installation directory, provides an application programming interface for building such a scheme.

To instruct SQL Remote to use your DLL for a particular message system, you must make a registry entry for that system. The registry entry should be made in the following location:

```

Software
  \Sybase
    \SQL Remote
      \message-system
        \encode_dll

```

where message-system is one of the SQL Remote message systems (**file**, **smtp**, and so on). You should set this registry entry to the name of your encoding DLL.

### Encoding and decoding must be compatible

If you implement a custom encoding, you must make sure that the DLL is present at the receiving end, and that the DLL is in place to decode your messages properly.

## The message tracking system

SQL Remote has a message tracking system to ensure that all replicated operations are applied in the correct order, no operations are missed, and no operation is applied twice.

Message system failures may lead to replication messages not reaching their destination, or reaching it in a corrupt state. Also, messages may arrive at their destination in a different order from that in which they were sent. This section describes the SQL Remote system for detecting and correcting message system errors, and for ensuring correct application of messages.

If you are using an e-mail message system, you should confirm that e-mail is working properly between the two machines if SQL Remote messages are not being sent and received properly.

The SQL Remote message tracking system is based on status information maintained in the **remotouser** SQL Remote system table. The table is maintained by the Message Agent. The Message Agent at a subscriber database sends confirmation to the publisher database to ensure that **remotouser** is maintained properly at each end of the subscription.

For Adaptive Server Anywhere, the **remotouser** table is the **sys.sysremotouser** system table. For Adaptive Server Enterprise, this is the **sr\_remotouser** table.

### Status information in the remotouser table

The **remotouser** SQL Remote system table contains a row for each subscriber, with status information for messages sent to and received by that subscriber. At the consolidated database, **remotouser** contains a row for each remote user. At each remote database, **remotouser** contains a single row maintaining information for the consolidated database. (Recall that the consolidated database subscribes to publications from the remote database.)

The **remotouser** SQL Remote system table at each end of a subscription is maintained by the Message Agent.

### Tracking messages by transaction log offsets

The message-tracking status information takes the form of offsets in the transaction logs of the publisher and subscriber databases. Each COMMIT is marked in the transaction log by a well-defined offset. The order of transactions can be determined by comparing their offset values.

---

Message ordering	When messages are sent, they are ordered by the offset of the last COMMIT of the preceding message. If a transaction spans several messages, there is a serial number within the transaction to order the messages correctly. The default maximum message size is 50,000 bytes, but you can use the Message Agent <code>-l</code> command-line switch to change this setting.
Sending messages	The <b>log_send</b> column holds the local transaction log offset for the latest message sent to the subscriber. When the Message Agent sends a message, it sets the <b>log_send</b> value to the offset of the last COMMIT in the message. Once the message has been received and applied at the subscribed database, confirmation is sent back to the publisher. When the publisher Message Agent receives the confirmation, it sets the <b>confirm_send</b> column for that subscriber with the local transaction log offset. Both <b>log_send</b> and <b>confirm_send</b> are offsets in the local database transaction log, and <b>confirm_send</b> cannot be a later offset than <b>log_send</b> .
Receiving messages	When the Message Agent at a subscriber database receives and applies a replication update, it updates the <b>log_received</b> column with the offset of the last COMMIT in the message. The <b>log_received</b> column at any subscriber database therefore contains a transaction log offset in the publisher database's transaction log. After the operations have been received and applied, the Message Agent sends confirmation back to the publisher database and also sets the <b>confirm_received</b> value in the local SYSREMOTEUSER table. The <b>confirm_received</b> column at any subscriber database contains a transaction log offset in the publisher database's transaction log.
Subscriptions are two-way	SQL Remote subscriptions are two-way operations: each remote database is a subscriber to publications of the consolidated database and the consolidated database subscribes to a matching publication from each remote database. Therefore, the <b>remoteuser</b> SQL Remote system tables at the consolidated and remote database hold complementary information.  The Message Agent applies transactions and updates the <b>log_received</b> value atomically. If a message contains several transactions, and a failure occurs while a message is being applied, the <b>log_received</b> value corresponds exactly to what has been applied and committed.
Resending messages	The <b>remoteuser</b> SQL Remote table contains two other columns that handle resending messages. The <b>resend_count</b> and <b>rereceive_count</b> columns are retry counts that are incremented when messages get lost or deleted for some reason.  The <b>log_send</b> column is updated by the Message Agent based on the SEND frequency information (SEND AT or SEND EVERY as specified in the GRANT REMOTE statement or <b>sp_grant_remote</b> procedure). The value of <b>log_send</b> must be greater than that of <b>log_send</b> for a user in order for messages to be sent to that user.

## Handling of lost or corrupt messages

When messages are received at a subscriber database, the Message Agent applies them in the correct order (determined from the log offsets) and sends confirmation to the publisher. If a message is missing, the Message Agent increments the local value of **rereceive\_count**, and requests that it be resent. Other messages present or en route are not applied.

The request from a subscriber to resend a message increments the **resend\_count** value at the publisher database, and also sets the publisher's **log\_sent** value to the value of **confirm\_sent**. This resetting of the **log\_sent** value causes operations to be resent.

### **Users cannot reset log\_sent**

The **log\_sent** value cannot be reset by a user, as it is in a system table.

### Message identification

Each message is identified by three values:

- ◆ Its **resend\_count**.
- ◆ The transaction log offset of the last COMMIT in the previous message.
- ◆ A serial number within transactions, for transactions that span messages.

Messages with a **resend\_count** value smaller than **rereceive\_count** are not applied; they are deleted. This ensures that operations are not applied more than once.