### How to Install SQL Server 2000

This set of How To topics includes common procedures used in installing Microsoft® SQL Server<sup>™</sup> 2000.

### How to install SQL Server 2000 (Setup)

#### To install SQL Server 2000

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, click **SQL Server 2000 Prerequisites**, and then click **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 4. In the **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.

For a remote installation, click **Remote Computer**. You can then type a computer name or click **Browse** to locate a remote computer.

If a cluster is detected, **Virtual server** is the default option.

- 5. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 6. Follow directions on the **User Information**, **Software License Agreement** and related screens.
- 7. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 8. In the **Instance Name** dialog box, if the **Default** check box is

available, you can install either the default or a named instance. If the **Default** check box is not available, a default instance has already been installed, and you can install only a named instance.

- To install the default instance, select the **Default** check box, and click **Next**.
- To install a named instance, clear the **Default** check box, and type a new named instance in the **Instance Name** edit box. Click **Next**.
- 9. In the **Setup Type** dialog box, click **Typical** or **Minimum**, and then click **Next**.

If you want to select components and subcomponents, change character set, network libraries or other settings, click **Custom**, and then click **Next**.

10. In the **Service Accounts** dialog box, accept the default settings, enter your domain password, and then click **Next**.

For information about services account options, see <u>Services</u> <u>Accounts</u>.

11. In the **Authentication Mode** dialog box, accept the default setting, and click **Next**.

To use Mixed Mode, see <u>Authentication Modes</u>.

- 12. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.
- 13. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.

Click **Help** for information about licensing or see your system administrator.

14. In the Setup Complete dialog box, click Yes, I want to restart my

**computer now**, and then click **Finish**.

#### See Also

- How to add components to an instance of SQL Server 2000 (Setup)
- How to create a case-sensitive instance of SQL Server (Setup)

How to install a named instance of SQL Server (Setup)

### How to install client tools only (Setup)

You can install client tools only using any SQL Server compact disc, on any supported operating system. For more information, see <u>How to install tools only</u> <u>from any compact disc</u>.

#### To install client tools only for SQL Server 2000

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 5. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 6. In the **Installation Definition** dialog box, click **Client tools only**, and then click **Next**.
- 7. In the **Select Components** dialog box, accept the defaults or select the components you want, and then click **Next**.

You can select an item in the **Components** list, such as **Management Tools**, and then select items from the related **Sub-Components** list, such as **Enterprise Manager**. Click to select items you want to install; clear the check box of the items you do not want to install.

For information about each component, select the item, and view the Description box.

8. In the **Start Copying Files** dialog box, click **Next** to complete the installation of the client tools.

# How to install tools only from any compact disc (Setup)

**Note** In this procedure, you can use the installation disc for any edition of SQL Server 2000 on a computer with any of the operating systems supported by SQL Server 2000.

#### To install tools only from any compact disc

- 1. Insert a Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. This can be the installation disc for any edition of SQL Server 2000, without regard to operating system support. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 5. In the **Select Components** dialog box, accept the defaults or select the components you want, and then click **Next**.

You can select an item in the **Components** list, such as **Management Tools**, and then select items from the related **Sub-Components** list, such as **Enterprise Manager**. Click to select items you want to install; clear the check box of the items you do not want to install.

For information about each component, select the item, and view the

Description box.

6. In the **Start Copying Files** dialog box, click **Next** to complete the installation of the client tools.

### How to install connectivity only (Setup)

The connectivity-only option installs Network Libraries and MDAC (Microsoft® Data Access Components).

#### To install connectivity only for SQL Server 2000

- 1. Insert the Microsoft SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, click **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 4. In the **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 5. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 6. Follow the directions on the **User Information, Software License Agreement** and related screens.
- 7. In the **Installation Definition** dialog box, click **Connectivity Only**, and then click **Next**.
- 8. In the **Start Copying Files** dialog box, click **Next** to complete the installation.

# How to install a named instance of SQL Server 2000 (Setup)

You can install a named instance of Microsoft® SQL Server<sup>™</sup> 2000 the first time you run SQL Server Setup or later after the default instance is installed. For each additional named instance you want to install, follow this procedure.

**Note** If you have a SQL Server 7.0 installation on your computer, the installation remains intact during the installation of a named instance of SQL Server 2000. A default instance of SQL Server 2000 will overwrite a SQL Server 7.0 installation (as the previous default installation), but a named instance does not overwrite SQL Server 7.0.

#### To install a named instance of SQL Server 2000

- 1. Insert the SQL Server 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.

If you are running Microsoft Windows® 95, **Install Common Controls Library Update.** 

- 3. Select **Install Database Server** and setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**. In the **Computer Name** dialog box, **Local Computer** is the default option, and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server, or install Client Tools,** and then click **Next**.
  - If this is the first SQL Server 2000 installation on your computer, follow the directions on the **User Information**, **Software License Agreement**, and related screens.

- If an installation of SQL Server 2000 exists on your computer, these screens are omitted.
- 5. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 6. In the **Instance Name** dialog box, clear the **Default** check box, and type a name for the new named instance, and then click **Next**.

**Note** If you have an existing default installation (either SQL Server 7.0 or 2000), the Default check box is not available.

If you have typed an instance name, and later return to the Instance Name dialog box to change the name before completing setup, you can do so. However, a workaround is necessary to edit the instance name box, which will be unavailable after clicking Back to get to this dialog box. Select the Default checkbox, then immediately clear it, and you will be able to edit the instance name.

For more information about instance names, click Help.

7. In the **Setup Type** dialog box, select **Typical**, **Minimum**, or **Custom**, and then click **Next**.

If you want to select subcomponents or change character set, network libraries, or other settings, click **Custom**.

8. In the **Service Accounts** dialog box, accept the default settings, enter your domain password, and then click **Next**.

For information about services account options, see <u>Services</u> <u>Accounts</u>.

9. In the **Authentication Mode** dialog box, accept the default setting, and click **Next**.

To use **Mixed Mode** authentication, see <u>Authentication Modes</u>.

10. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.

11. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.

Click **Help** for information about licensing, or see your system administrator.

12. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish**.

#### See Also

Working with Named and Multiple Instances of SQL Server 2000

Running SQL Server 7.0 Along with a Named Instance of SQL Server 2000

## How to upgrade a SQL Server 7.0 installation to SQL Server 2000 (Setup)

**CAUTION** This version upgrade procedure overwrites your Microsoft® SQL Server<sup>TM</sup> 7.0 installation; the installation no longer exists on your computer. In addition, previous registry settings are removed. For example, after upgrading you will need to re-register your servers.

To restore the SQL Server 7.0 installation, you must first uninstall SQL Server 2000, perform a complete reinstall of the SQL Server 7.0 files, and then restore your backed-up SQL Server 7.0 databases.

#### To upgrade SQL Server 7.0 to SQL Server 2000

1. Insert the Microsoft SQL Server 2000 compact disc for the edition to which you want to upgrade into your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.

**Note** If you have purchased an edition of SQL Server with more features than your current SQL Server 7.0 installation, the upgrade process will perform both the version and edition upgrade at the same time.

- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 4. In the **Installation Selection** dialog box, click **Upgrade, remove, or add components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, **Default** will be selected. Click

#### Next.

**Note** When upgrading, SQL Server 7.0 automatically becomes the default instance of SQL Server 2000.

- 6. In the **Existing Installation** dialog box, click **Upgrade your existing installation**, and then click **Next**.
- 7. In the Upgrade dialog box, you are prompted as to whether you want to proceed with the requested upgrade. Click Yes, upgrade my <text specific to the upgrade> to start the upgrade process, and then click Next. The upgrade runs until finished.
- 8. In the **Connect to Server** dialog box, select an authentication mode, and then click **Next**.

If you are not sure which mode to use, accept the default: **The Windows account information I use to log on to my computer with** (Windows).

- 9. In Start Copying Files dialog box, click Next.
- 10. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish**.

#### See Also

Authentication Modes

How to perform an edition upgrade within SQL Server 2000 (Setup)

### How to upgrade databases online using the Copy Database Wizard (Enterprise Manager)

To upgrade a SQL Server 7.0 database to a SQL Server 2000 database

- 1. Expand a server group, and then expand a server.
- 2. Right-click the server, point to **All Tasks**, and then click **Copy Database Wizard**.
- 3. Complete the steps in the wizard.

**IMPORTANT** After upgrading databases from SQL Server 7.0, run sp\_updatestats (update statistics) against the database on the destination server to ensure optimal performance of the copied database.

#### See Also

Copy Database Wizard Help

Database Upgrade from SQL Server 7.0 (Copy Database Wizard)

# How to perform an edition upgrade within SQL Server 2000 (Setup)

### To upgrade a SQL Server 2000 installation to a different edition of SQL Server 2000

- 1. Insert the Microsoft® SQL Server 2000<sup>™</sup> compact disc for the edition you want to install into your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, select **Local Computer** or **Remote computer**.
- 4. In the **Installation Selection** dialog box, click **Upgrade**, **Remove**, **or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, click **Next**.
- 6. In the **Existing Installation** dialog box, click **Upgrade your existing installation**, and then click **Next**.
- If Setup detects that you are doing an edition upgrade, the Upgrade dialog box appears. Click Yes, Upgrade my <text specific to the upgrade> to upgrade the feature set of your current installation, and click Next.

8. After the upgrade is completed, you are prompted as to whether you want to install additional components. If you click **Yes**, the **Select Components** dialog box appears. Accept the defaults or select the additional components you want to install, and then click **Next**.

You can select an item in the **Components** list, and then select items from the related **Sub-Components** list. Click to select items you want to install; clear the check box of the items you do not want to install.

- 9. When you are finished specifying options, in the **Start Copying Files** dialog box, click **Next**.
- 10. In the **Setup Complete** dialog box, click **Yes, I want to restart my computer now**, and then click **Finish** to complete the edition upgrade.

#### See Also

Upgrading an Existing Installation of SQL Server

# How to uninstall an existing installation of SQL Server (Setup)

## To uninstall an existing installation of SQL Server 7.0 or SQL Server 2000 (default or named instance)

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then setup prepares the SQL Server Installation Wizard. At the **Welcome** screen, click **Next**.
- 3. In **Computer Name** dialog box, select **Local Computer** or **Remote computer**.
- 4. In the **Installation Selection** dialog box, click **Upgrade, Remove, or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 5. In the **Instance Name** dialog box, **Default** is selected if you have the Default instance installed. If you want to uninstall a named instance, select it from the Instance Name list box, and then click **Next**.
- 6. In the **Existing Installation** dialog box, click **Uninstall your existing installation**, and then click **Next**.
- 7. Setup removes the selected installation. In the **Uninstalling** dialog box, click **Next**, and then in the **Setup Complete** dialog box, click **Finish**.

### How to test an installation of SQL Server 2000 (Command Prompt)

#### To test the installation

1. Start Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 by entering from a command prompt:

For the default instance, use:

net start mssqlserver

For a named instance, include the instance name, for example:

net start MSSQL\$Instance1

2. Connect to SQL Server by entering:

For the default instance, use:

osql /Usa /P <administrator password>

For a named instance, include both the server and instance name, for example:

osql /Usa /P /S Machine1\Instance1

When **osql** connects, this **osql** prompt appears:

1>

If **osql** cannot connect, an ODBC error is returned.

 Enter a simple query, such as: SELECT @@SERVERNAME GO

The **osql** utility returns the server name:

1> SELECT @@SERVERNAME 2> GO

-----

WOLFHOUND

(1 row affected) 1>

 4. Verify that you have checked a SQL Server 2000 server by entering: SELECT @@VERSION GO

The **osql** utility returns the version information.

5. Quit the **osql** utility by entering: Exit

## How to change SQL Server services login account information (Windows NT)

#### To change SQL Server services login account information (Windows NT)

- 1. On the **Start** menu, point to **Settings**, and then click **Control Panel**.
- 2. Double-click **Services**.
- 3. In the **Services** dialog box, double-click **MSSQLSERVER** in the Service list.

**Note** For named instances, the instance name is included. For example, to modify the user account for **Instance1**, you double-click **MSSQL\$Instance1**.

- 4. In the **Service** dialog box, under **Log on as**, select **This account**, and then enter the changed account information.
- 5. Repeat Steps 3 and 4 above for SQL Server Agent. In the Services dialog box, double-click SQLSERVERAGENT (or SQLAgent\$Instance1 for a named instance), and then enter the changed account information in the Service dialog box.
- 6. Start SQL Server Enterprise Manager, and change the user account information there, as well, for both SQL Server and SQL Server Agent For more information see <u>How to change SQL Server services login</u> <u>account information (Enterprise Manager)</u>.

## How to change SQL Server services login account information (Windows)

To change SQL Server services login account information (Windows 2000)

- 1. On the **Start** menu, point to **Programs/Administrative Tools**, and then click **Services**.
- 2. Right-click **MSSQLServer**, and then click **Properties**.
- 3. On the **Log On** tab, enter and confirm the new password, and then restart services using the SQL Server Service Manager.
- 4. Repeat the password reset for SQLServerAgent and other services.
- 5. Start SQL Server Enterprise Manager, and change user account information there, as well, for both SQL Server and SQL Server Agent For more information, see <u>How to change SQL Server services login</u> <u>account information (Enterprise Manager)</u>.

# How to change SQL Server services login account information (Enterprise Manager)

**Note** If you are running Microsoft® Windows® 2000 and want to use the Windows 2000 Encrypted File System to encrypt any Microsoft SQL Server<sup>™</sup> files, you must unencrypt the files before you can change the SQL Server service accounts. If you do not unencrypt the files and then reset the SQL Server service accounts, you cannot unencrypt the files.

#### To change the MSSQLServer service login (Enterprise Manager)

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. In the SQL Server Properties dialog box, click the **Security** tab.
- 4. In the **Startup service account** box, the option for **This Account** is selected, indicating that the SQL Server service account is a Windows domain account. Enter changes as necessary for the account and password.

#### To change the SQLServerAgent service login (Enterprise Manager)

**Note** You can change the SQLServerAgent service account to a non Microsoft Windows NT® 4.0 administrator account. However, the Windows NT 4.0 account must be a member of the **sysadmin** fixed server role to run SQL Server Agent.

- 1. Expand a server group, and then expand a server.
- 2. Expand Management.
- 3. Right-click **SQL Server Agent**, and then click **Properties**.

- 4. In the **SQL Server Agent Properties** dialog box, click the **General** tab.
- 5. In the **Service startup account** box, enter the appropriate account and password.

#### See Also

Creating SQL Server Services User Accounts

**Changing Passwords and User Accounts** 

### How to rebuild the registry (Setup)

#### To rebuild the registry

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In the **Computer Name** dialog box, click **Next**.
- 4. In the **Installation Selection** dialog box, click **Advanced options**, and then in the **Advanced Options** dialog box, click **Registry Rebuild**. Click **Next**.
- 5. A message appears informing you that Setup rebuilds the registry based on information you supply in the subsequent screens.

**CAUTION** The setup options you enter must be the same choices that you entered during the initial installation. If you do not know or are not sure of this information, do not use this registry rebuild process. Instead, you must uninstall and reinstall SQL Server to restore the registry.

6. To prepare for the registry rebuild, enter the same information and options that you entered during the initial installation of SQL Server in the setup screens as they appear. When you have finished, the registry rebuild will occur.

**Note** Rebuilding the registry includes re-copying external components such as MDAC and MS DTC.

# How to rebuild the master database (Rebuild Master utility)

#### To rebuild the master database

- 1. Shutdown Microsoft® SQL Server<sup>™</sup> 2000, and then run Rebuildm.exe. This is located in the Program Files\Microsoft SQL Server\80\Tools\Binn directory.
- 2. In the **Rebuild Master** dialog box, click **Browse**.
- 3. In the **Browse for Folder** dialog box, select the \Data folder on the SQL Server 2000 compact disc or in the shared network directory from which SQL Server 2000 was installed, and then click **OK**.
- 4. Click **Settings**. In the **Collation Settings** dialog box, verify or change settings used for the **master** database and all other databases.

Initially, the default collation settings are shown, but these may not match the collation selected during setup. You can select the same settings used during setup or select new collation settings. When done, click **OK**.

5. In the **Rebuild Master** dialog box, click **Rebuild** to start the process.

The Rebuild Master utility reinstalls the **master** database.

**Note** To continue, you may need to stop a server that is running.

#### See Also

Collation Settings in Setup

# How to perform a remote installation of SQL Server 2000 (Setup)

#### To perform a remote installation

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In **Computer Name** dialog box, click **Remote Computer**. You can then type a computer name or click **Browse** to locate a remote computer.
- 4. In the **Installation Selection** dialog box, click **Create a new instance of SQL Server,** or **install Client Tools**.
- 5. Follow the directions on the **User Information**, Software License Agreement, and related screens.
- 6. In the **Remote Setup Information** dialog box, enter password and other information. For more information, see <u>Remote Setup</u> <u>Information</u>. After you finish defining options, click **Next**.
- 7. In the **Installation Definition**, **Instance Name**, **Setup Type**, and subsequent setup screens, select the options you want for the remote installation.

SQL Server Setup creates the Setup.iss file in your local system folder with the options you have specified.

- 8. After Setup creates Setup.iss, the **Setup Complete** dialog box appears. Click **Finish** to start the remote installation process.
- 9. When the process is finished, click **OK** in the message box that appears. Reboot the remote computer before running the remote instance.

### How to record an unattended installation file (Setup)

The **Record Unattended** Setup option allows you to simulate an installation and create an .iss file that can be used later for an unattended installation of Microsoft® SQL Server<sup>™</sup> 2000. SQL Server files are not installed in this process.

#### To create a file for an unattended installation

- 1. Insert the Microsoft SQL Server 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 3. In the **Computer Name** dialog box, select the option you want, and click **Next**.
- 4. In the **Installation Selection** dialog box, click **Advanced options**, and then in the **Advanced Options** dialog box, click **Record Unattended .ISS file.** Click **Next**.
- 5. In subsequent Setup screens, select the options you want for the unattended installation. After you finish selecting the options, in the **Setup Information** screen, click **Next**.
- 6. In the **Setup Complete** screen, click **Finish**.

This message appears: "Setup has collected the information needed to create an unattended installation file (.iss) for use with later unattended installations of SQL Server."

SQL Server Setup then creates the Setup.iss file in the %windir%

location with the options you have specified.

To run the file, see <u>How to run an unattended installation of SQL</u> <u>Server 2000 (Command Prompt)</u>.

### See Also

Performing an Unattended Installation

# How to run an unattended installation of SQL Server 2000 (Command Prompt)

You can run an unattended installation by using sample batch files and setup initialization files included on the Microsoft® SQL Server<sup>™</sup> 2000 compact disc. Or, you can run the Setup program directly from the command prompt in the appropriate directory for the edition of SQL Server you want to install, using arguments as needed.

#### To run an unattended installation using ready-made batch files

- 1. Locate the .bat and .iss files in the root directory of your SQL Server 2000 compact disc.
- 2. View the .bat and associated .iss files, and modify if necessary. For more information, see <u>Creating a Specialized Setup File</u>.
- 3. Run the appropriate batch and setup files from the command prompt:
  - For a standard unattended installation, run Sqlins.bat.
  - For a client-only unattended installation, run Sqlcli.bat.
  - For a custom unattended installation, run Sqlcst.bat.

#### To run an unattended installation directly from the command prompt

- 1. Run Setupsql.exe from the Setup directory in the appropriate architecture directory.
- 2. Use arguments as needed:
- -f1 <initialization file path>
   Selects an unattended setup initialization file.

start /wait command (with the -SMS option)

Returns control to the command prompt only after SQL Server Setup completes.

-s flag

Causes the Setup program to run in silent mode with no user interface.

For examples of command prompt options and arguments, see the sample .bat files on your SQL Server 2000 compact disc.

#### See Also

Performing an Unattended Installation

How to record an unattended installation file (Setup)

# How to add components to an instance of SQL Server 2000 (Setup)

**Note** You cannot remove components by clearing checkboxes in the **Select Components** dialog box. If you need to remove components from an instance of SQL Server, you must uninstall the instance.

#### To add components to an instance (default or named) of SQL Server 2000

- 1. Run SQL Server Setup, select **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 2. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 3. In the **Installation Selection** dialog box, click **Upgrade, Remove, or Add Components to an existing instance of SQL Server**, and then click **Next**.
- 4. In the **Instance Name** dialog box, **Default** is selected if you have the Default instance installed. If you want to add components to a named instance, select it from the Instance Name list, and then click **Next**.
- 5. In the **Existing Installation** dialog box, click **Add Components to your existing installation**, and then click **Next**.
- In the Select Components dialog box, select a component from the Components list, and then select items from the related Sub-Components list. Click to select items you want to add, and then click Next.

For information about each component, select the item, and view the **Description** box.

7. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box to add components to the selected instance of SQL Server.

### See Also

How to uninstall an existing installation of SQL Server (Setup)

# How to access SQL Server Books Online for SQL Server 7.0

If you have Microsoft® SQL Server<sup>™</sup> 7.0 running as the default instance (and SQL Server 2000 as a named instance), SQL Server Books Online for SQL Server 7.0 remains intact on your computer. You can access SQL Server Books Online from the **Start** menu or create a shortcut to it on your desktop.

#### To access SQL Server Books Online for SQL Server 7.0 from the Start menu

• On the **Start** menu, point to **Programs** and **Microsoft SQL Server 7.0**, and then click **Books Online**.

#### To create a shortcut to SQL Server Books Online for SQL Server 7.0

- 1. Locate Sqlbol.chm on your computer. (The default location is C:\Mssql7\Books.)
- 2. Right-click Sqlbol.chm, and then click **Create Shortcut**.
- 3. Copy the shortcut to your desktop, where you can use it to access SQL Server Books Online.

To install SQL Server Books Online for SQL Server 7.0 for the first time, or to reinstall it, you must install it from the SQL Server 7.0 compact disc or the SQL Server Web site.

### **To reinstall SQL Server Books Online for SQL Server 7.0 from the SQL Server 7.0 compact disc**

- 1. Insert the Microsoft SQL Server 7.0 compact disc in your CD-ROM drive.
- 2. Locate the file Sqlbol.chm on the compact disc, and copy it to a location on your computer.

3. Create a shortcut on your desktop to SQL Server Books Online for SQL Server 7.0.

### **To download SQL Server Books Online for SQL Server 7.0 from the SQL Server Web site**

- 1. Go to the Microsoft SQL Server Web site, at Microsoft Web site.
- 2. On the SQL Server Welcome page, click **Support**.
- 3. On the Support page, click **Documentation** and follow instructions to access SQL Server 7.0 Books Online.

### How to install English Query (Setup)

#### **To install English Query**

1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.

#### 2. Select SQL Server 2000 Components.

3. On the Install Components screen, select **Install English Query**.

No further selections are necessary. SQL Server Setup installs English Query on your computer.

### How to install Analysis Services (Setup)

The following procedure is a shortened version of the steps to install Analysis Services. For a more complete installation procedure and for related information, see <u>Running Setup</u>.

#### **To install Analysis Services**

- 1. Insert the Microsoft® SQL Server<sup>™</sup> 2000 compact disc in your CD-ROM drive. If the compact disc does not autorun, double-click Autorun.exe in the root directory of the compact disc.
- 2. Select SQL Server 2000 Components.
- 3. On the Install Components screen, select **Install Analysis Services**.
- 4. At the Welcome screen for Microsoft SQL Server 2000 Analysis Services, click **Next**.
- 5. Follow the directions on the **User Information**, **Software License Agreement**, and related screens.
- 6. In the **Select Components** dialog box, select or clear components as needed, and then click **Next**.

If you want to change the default location of the Analysis Services program files, click **Browse** at **Destination Folder** and select a folder location.

- 7. In the **Data Folder Location** dialog box, accept or change the default location for data files, and then click **Next**.
- 8. In the **Select Program Folder** dialog box, accept or change the default settings, and then click **Next**.

SQL Server Setup installs Analysis Services on your computer.

# How to create a case-sensitive instance of SQL Server 2000 (Setup)

#### To create a case-sensitive instance of SQL Server 2000

- 1. Run SQL Server Setup to install **SQL Server 2000 Components**, select **Install Database Server**, and then click **Next** at the Welcome screen of the SQL Server Installation Wizard.
- 2. In **Computer Name** dialog box, **Local Computer** is the default option and the local computer name appears in the edit box. Click **Next**.
- 3. In the **Installation Selection** dialog box, click click **Create a new instance of SQL Server, or install Client Tools**, and then click **Next**.
- 4. Follow the directions on the **User Information** and related screens.
- 5. In the **Installation Definition** dialog box, click **Server and Client Tools**, and then click **Next**.
- 6. In the **Instance Name** dialog box:
  - To create a case-sensitive default instance, accept the **Default** check box and click **Next**.
  - To create a case-sensitive named instance, clear the **Default** check box and type an instance name.
- 7. In the **Setup Type** dialog box, click **Custom**, and click **Next**.
- 8. In the **Select Components**, **Services Accounts**, and **Authentication Mode** dialog boxes, change or accept the default settings, and then click **Next**.

- 9. In the **Collation Settings** dialog box, you have two options:
  - To make a Windows Locale collation case-sensitive, select **Collation designator** and then select the correct collation designator from the list. Clear the **Binary** check box, and then select the **Case-sensitive** check box.
  - To make a SQL collation case-sensitive, select **SQL Collations**, and then select the correct collation name.

For more information about collation options, click **Help**. When you finish setting the options, click **Next**.

- 10. In subsequent dialog boxes, change or accept the default settings, and then click **Next**.
- 11. When you are finished specifying options, click **Next** in the **Start Copying Files** dialog box.
- 12. In the **Choose Licensing Mode** dialog box, make selections according to your license agreement, and click **Continue** to begin the installation.
- 3. Click **Help** for information about licensing, or see your system administrator.

#### See Also

**Collation Settings in Setup** 

### How to set client code pages

To set client code pages under the Windows NT, Windows 98, or Windows 2000 operating systems

• Use the Regional Settings application in Control Panel as described in the Microsoft® Windows NT®, Microsoft Windows® 98, or Microsoft Windows 2000 documentation.

# How to switch from SQL Server 6.5 to SQL Server 2000 (Command Prompt)

#### To switch from SQL Server 6.5 to SQL Server 2000

• Run Vswitch.exe.

#### -SwitchTo <65|80>

Determines which version of Microsoft® SQL Server<sup>™</sup> 2000 to activate.

#### -Silent <0|1>

Determines if any user interface or messages are displayed. If **1** is specified, a user interface or messages are not displayed. The default is **0**.

#### Examples

c:\...\vswitch -SwitchTo 80 -Silent 1

# How to switch from SQL Server 6.5 to SQL Server 2000 (Windows)

#### To switch from SQL Server 6.5 to SQL Server 2000

• On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, and then click **Microsoft SQL Server 2000**.

SQL Server Setup switches from Microsoft® SQL Server<sup>™</sup> 2000 version 6.5 to SQL Server 2000.

### How to remove SQL Server 2000 (Windows)

You can remove instances of Microsoft® SQL Server<sup>™</sup> 2000 using Control Panel. Each named instance must be removed separately. When upgrading or maintaining instances, you can remove SQL Server using the Uninstall option in Setup. For more information, see <u>How to uninstall an existing installation</u> (Setup).

You cannot remove a selected component of SQL Server 2000 after it is installed. To remove components, you must remove the entire instance.

#### To remove a named instance of SQL Server 2000

- 1. In **Control Panel**, click **Add/Remove** programs.
- 2. Select a name of an instance of SQL Server 2000, and click **Remove**.

#### To remove all instances of SQL Server 2000

- 1. In **Control Panel**, click **Add/Remove** programs.
- 2. Repeat the removal process for each instance of SQL Server 2000 that is installed.

SQL Server 2000 is uninstalled, but some files may remain. Manually delete directories if any files related to SQL Server 2000 still exist.

#### See Also

**Directories and File Locations** 

### **How To Upgrade from SQL Server 6.5**

The How To topics in this section are specific to the process of converting data from Microsoft® SQL Server<sup>™</sup> 6.5 to Microsoft SQL Server 2000 using the SQL Server Upgrade Wizard.

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft SQL Server 2000 already installed on your computer.

#### **Considerations when upgrading from SQL Server 6.5:**

- During the upgrade process, the SQL Server 6.5 server is stopped and started while objects are scripted and data is extracted. When the data transfer starts, only SQL Server 2000 is running, and it is not possible to access SQL Server 6.5.
- If you are upgrading your existing SQL Server 6.5 server to a different computer that is running SQL Server 2000, both computers should be configured to use a domain user name and password for the MSSQLServer service.
- During this upgrade, user-defined messages created in SQL Server 6.5 using **sp\_addmessage** are not converted to SQL Server 2000. To retain these custom messages, manually copy the messages added in SQL Server 6.5 to your installation of SQL Server 2000.

#### See Also

Troubleshooting the SQL Server Upgrade Wizard

Completing the SQL Server Upgrade Wizard

**Upgrade Log Files** 

Upgrading to SQL Server 2000 FAQ

#### How to change the size of tempdb in SQL Server 6.5 (ISQL/w)

#### To change the size of tempdb in SQL Server 6.5

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server 6.5**, and then click **ISQL/w**.
- 2. Enter the **sa** password, and then click **Connect**.
- 3. Execute a DISK INIT command to increase the size of the **tempdb** device to at least 25 MB.
- 4. Execute an ALTER DATABASE command to increase the size of the **tempdb** database to at least 25 MB.

#### Examples

--Increase the size of the tempdb device DISK INIT name = 'tempdb1',physname = 'c:\mssql\data\tempdb1.DA' GO --Increase the size of tempdb

ALTER DATABASE tempdb ON tempdb1 = 25

### How to change to the current server name in the SQL Server 6.5 master database (ISQL/w)

### To change to the current server name in the SQL Server 6.5 master database

- Start Microsoft<sup>®</sup> SQL Server<sup>™</sup> in minimal configuration mode. In a command prompt window, from the \Mssql\Binn directory, run: sqlservr -f
- 2. On the **Start** menu, point to **Programs** /**Microsoft SQL Server 6.5**, and then click **ISQL**/w.
- 3. Enter the **sa** password, and then click **Connect**.
- 4. Execute SELECT @@SERVERNAME to retrieve the former server name.
- 5. Execute **sp\_dropserver** to drop the former server.
- 6. Execute **sp\_addserver** to add the current server.
- 7. Stop SQL Server. In the command prompt window, press Ctrl+C.
- 8. Restart SQL Server.
- 9. Execute SELECT @@SERVERNAME to verify the current server name.

#### Examples

--Start SQL Server in minimal configuration mode.

--Retrieve the former server name.

SELECT @@SERVERNAME

--Drop the server returned from the previous select.

sp\_dropserver 'SERVER6X'

--Add the current server.

sp\_addserver 'SERVER70', local

--Stop SQL Server.

--Restart SQL Server in minimal configuration mode.

--Verify the current server name.

SELECT @@SERVERNAME

## How to update the device file locations in the SQL Server 6.5 master database (ISQL/w)

To update the device file locations in the SQL Server 6.5 master database

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server 6.5**, and then click **ISQL/w**.
- 2. Enter the **sa** password, and then click Connect.
- 3. Select from **sysdevices** in the **master** database to view the old device file locations.
- 4. Execute **sp\_configure** to allow updates to the system tables, and then reconfigure with override.
- 5. Update the device file locations that have changed.
- 6. Execute **sp\_configure** to disallow updates to the system tables, and then reconfigure with override.

#### Examples

--View the old device file locations SELECT phyname FROM sysdevices

--Allow updates to the system tables sp\_configure 'allow updates',1 GO RECONFIGURE WITH OVERRIDE GO --Update device file locations that have changed **UPDATE** sysdevices

SET phyname = "E:\Data\HR\HR1.dat"

WHERE name = "HumanResources1"

GO

**UPDATE** sysdevices

SET phyname = "E:\Data\HR\HR1Log.dat"

WHERE name = "HumanResources1Log" GO

--Disallow updates to the system tables sp\_configure 'allow updates',0

GO

**RECONFIGURE WITH OVERRIDE** GO

#### How to estimate the disk space required for an upgrade from SQL Server version 6.5 to SQL Server 2000 (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

#### To estimate the disk space required for an upgrade

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. Select **Named pipe**; then click **Next**.
- 3. In **Export server (6.5)**, in the **Server name** box, enter the name of the local or remote computer on which SQL Server 6.5 resides.
- 4. In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**.

Unless you have changed it, the system administrator password for SQL Server 2000 is blank.

- 5. Include the databases to upgrade. Move any database not to include in the disk space estimation to the Exclude list, and then click **Next**.
- 6. Select **Use the default configuration or edit the default**; then click **Edit**.

The SQL Server Upgrade Wizard layout utility appears, showing the proposed layout of the SQL Server 2000 data files.

- 7. Click Advanced.
- 8. Click an object in the **Proposed database layout** box to view details

in the **Object details** box.

- 9. The **Drive summary** box shows the estimated size of all SQL Server 2000 data files and the free disk space left on all of the local fixed disks. On the **Options** menu, select **Freespace includes 6.5 files** to view the free space that would exist if the SQL Server 6.5 data files were deleted.
- 10. Click **Accept** to return to the **Database Creation** dialog box.
- 11. Click **Cancel** to quit the SQL Server Upgrade Wizard.

## How to edit the default database configuration (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

#### To edit the default database configuration

- 1. In the **Database Creation** dialog box of the SQL Server Upgrade Wizard, click **Edit**.
- 2. Click **Advanced** to view object details and drive summaries.
- 3. In the **Proposed database layout** box, double-click a database file.
- 4. Change any database file attributes, and then click **OK**.
- 5. View the changes to the drive summary.
- 6. When all changes have been made, click **Accept** to save the database configuration.

#### See Also

Proposed Database Layout

#### How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a direct pipeline (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

### To perform a SQL Server version 6.5 to SQL Server 2000 upgrade by named pipe

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. In the Data and Object Transfer screen, accept the default selections, including **Named pipe**, and then click **Next**. Verification options are recommended, but not required. Click **Help** for information.
- On the Logon screen, in the Server name box in the Export server (6.5) group box, enter the name of the local or remote computer on which Microsoft SQL Server version 6.5 is installed.

In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**. Unless you have changed it, the system administrator password for SQL Server 2000 is blank.

For **Import Server (2000)**, the server name is filled in. Enter the optional startup arguments, if you want. Click **Help** for information. When you are finished setting options, click **Next**.

- 4. In the message box asking if you want to continue, click **Yes** if you are ready to upgrade. The SQL Server Upgrade Wizard shuts down SQL Server 6.5 and starts SQL Server 2000.
- 5. In the **Code Page Selection** screen, accept or change the default settings, and then click **Next**.

6. In the **Database Selection** screen, include the databases to upgrade. Move any databases you do not want upgraded at this time to the **Exclude** list, and then click **Next**.

Converting all databases is recommended.

7. In the **Database Creation** dialog box, select **Use the default configuration or edit the default**, and then click **Next**.

Click **Edit** to examine and make changes to the proposed disk configuration within the layout utility. In the **Proposed Database Layout** box, make changes as needed. Click **Advanced** to view **Object Details** and **Drive Summary**. When you are finished, click **Accept** to return to the SQL Server Upgrade Wizard.

- 8. In the **System Configuration** screen, in **System objects to transfer**, select the object types to transfer from SQL Server 6.5 to SQL Server 2000:
  - Server configuration

Login and remote login registrations and server configuration options relevant to SQL Server 2000 are transferred as part of the version upgrade.

#### • Replication settings

All articles, subscriptions and publications of each selected database, plus the distribution database, if any, are transferred and upgraded.

#### • SQL Executive settings

All tasks scheduled by SQL Executive are transferred and upgraded so that SQL Server 2000 can schedule and run those tasks in SQL Server Agent.

- 9. In the System Configuration screen, in Advanced settings, for ANSI Nulls, select:
  - **Off**, if ANSI nulls should not be used when stored procedures are created. This is the default.

- **On**, if ANSI nulls should be used when stored procedures are created.
- 10. In **Quoted identifiers**, select one of these options, and then click **Next**:
  - **Mixed (or don't know)**, if some of your objects were created with QUOTED\_IDENTIFIER set to ON and others with it set to OFF, or if you are not sure how they were created.
  - **Off**, if all objects should be compiled with QUOTED\_IDENTIFIER set to OFF.
  - **On**, if all objects should be compiled with QUOTED\_IDENTIFIER set to ON.
- 11. In the **Completing the SQL Server Wizard** screen, view the summary of choices you have made. Click **View warnings and choices in notepad** to open a text version of the upgrade script. If all options are correct, click **Finish**.

The **SQL Server Upgrade Script Interpreter** screen appears, with information on the progress of the upgrade.

#### See Also

Order of Upgrade Using a Direct Pipeline or Tape Drive

#### How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)

**Note** To run the SQL Server Upgrade Wizard, you must have an instance of Microsoft® SQL Server<sup>™</sup> 2000 already installed on your computer.

### To perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server-Switch**, click **SQL Server Upgrade Wizard**, and then click **Next**.
- 2. In the **Data and Object Transfer** screen, click **Tape**, and then click **Next**. Verification options are recommended, but not required. Click **Help** for information.
- On the Logon screen, in the Server name box in the Export server (6.5) group box, enter the name of the computer on which Microsoft SQL Server version 6.5 is installed.

In the **Administrator password ('sa')** box, enter the **sa** password for SQL Server 6.5, and then click **Next**.

Unless you have changed it, the **sa** password for SQL Server 2000 is blank.

For **Import Server (2000)**, the server name is filled in. Enter optional startup arguments, if you want. Click **Help** for information. When you are finished setting options, click **Next**.

- 4. In the message box asking if you want to continue, click **Yes** if you are ready to upgrade. The SQL Server Upgrade Wizard switches to the SQL Server 2000 server.
- 5. In the Code Page Selection screen, accept or change the default settings, and then click **Next**.

- 6. In the Database Selection screen, include the databases to upgrade. Move any database not to be upgraded at this time to the Exclude list, and then click **Next**.
- 7. In **Device for data transfer**, specify the location of the tape drive.
- 8. In **6.5 device backup options**, select **Backup 6.5 devices before exporting data** if you have not backed up the databases already.

Prior to creating the SQL Server 2000 databases, the SQL Server Upgrade Wizard either prompts you to back up the SQL Server 6.5 devices or copies the devices for you automatically.

9. Select **Delete 6.5 devices before importing data** if necessary due to lack of disk space, and then click **Next**.

After objects and data are exported, and before creating databases in SQL Server 2000, the SQL Server Upgrade Wizard deletes the SQL Server 6.5 devices to reclaim disk space.

10. Select **Use the default configuration** or edit the default, and then click **Next**.

Click **Edit** to examine and make changes to the proposed disk configuration within the layout utility. In the **Proposed Database Layout** box, make changes as needed. Click **Advanced** to view **Object Details** and **Drive Summary**. When you are finished, click **Accept** to return to the SQL Server Upgrade Wizard.

- 11. In **System objects to transfer**, select the object types to transfer from SQL Server 6.5 to SQL Server 2000:
  - Server configuration

Login and remote login registrations and server configuration options relevant to SQL Server 2000 are transferred as part of the version upgrade.

#### • Replication settings

All articles, subscriptions, and publications of each selected database, plus the distribution database, if any, are transferred and upgraded.

#### • SQL Executive settings

All tasks scheduled by SQL Executive are transferred and upgraded so that SQL Server 2000 can schedule and run those tasks in SQL Server Agent.

- 12. In ANSI Nulls, select:
  - **Off**, if ANSI nulls should not be used when stored procedures are created. This is the default.
  - **On**, if ANSI nulls should be used when stored procedures are created.
- 13. In **Quoted Identifiers**, select one of these options, and then click **Next**:
  - **Mixed (or don't know)**, if some of your objects were created with QUOTED\_IDENTIFIER set to ON and others with it set to OFF, or if you are not sure how they were created.
  - **Off**, if all objects should be compiled with QUOTED\_IDENTIFIER set to OFF.
  - **On**, if all objects should be compiled with QUOTED\_IDENTIFIER set to ON.
- 14. In the **Completing the SQL Server Wizard** screen, view the summary of choices you have made. Click **View warnings and choices in notepad** to open a text version of the upgrade script. If all options are correct, click **Finish**.

The **SQL Server Upgrade Script Interpreter** screen appears with

information about the progress of the upgrade.

#### See Also

Order of Upgrade Using a Direct Pipeline or Tape Drive

### **SQL Server Enterprise Manager**

Microsoft® Management Console (MMC) is a tool that presents a common interface for managing different server applications in a Microsoft Windows® network. Server applications provide a component called an MMC snap-in that presents MMC users with a user interface for managing the server application. SQL Server Enterprise Manager is the Microsoft SQL Server<sup>TM</sup> MMC snap-in.

SQL Server Enterprise Manager is the primary administrative tool for SQL Server and provides an MMC-compliant user interface that allows users to:

- Define groups of SQL Server instances.
- Register individual servers in a group.
- Configure all SQL Server options for each registered server.
- Create and administer all SQL Server databases, objects, logins, users, and permissions in each registered server.
- Define and execute all SQL Server administrative tasks on each registered server.
- Design and test SQL statements, batches, and scripts interactively by invoking SQL Query Analyzer.
- Invoke the various wizards defined for SQL Server.

### **Administering SQL Server**

Microsoft® SQL Server<sup>™</sup> 2000 administration applications, and the accompanying services, are designed to assist the system administrator with all administrative tasks related to maintaining and monitoring server performance and activities.

Торіс	Description
Starting, Pausing, and	Explains how to start an instance of SQL
Stopping SQL Server	Server, and what you need to do before,
	during, and after you log in.
Failover Clustering	Describes how to set up and use a SQL
	Server 2000 failover cluster.
Importing and Exporting Data	Describes how to retrieve data from
	external sources and feed data to other
	applications.
Backing Up and Restoring	Describes how to protect and restore data
Databases	over a wide range of potential system
	problems.
Using the Copy Database	Describes how to copy or move databases
Wizard	between servers and upgrade databases
	from SQL Server version 7.0 to SQL Server
	2000.
Managing Servers	Describes how to register and configure
	remote and linked servers, add or remove
	servers, and modify server settings.
Managing Clients	Describes how to configure client
	connections with server components and
	change the default network protocol to meet
	the needs of your site.
Automating Administrative	Describes how to establish which
Tasks	administrative responsibilities will occur
	regularly, define jobs and alerts, and run
	SQL Server Agent.
Managing Security	Describes how to protect and safeguard

	database access by restricting permissions to include only authorized users.
Monitoring Server	Describes how to develop a strategy for
Performance and Activity	ensuring that server and activity
	performance are at acceptable levels.
Using the Web Assistant	Explains how to use the wizard to create
<u>Wizard</u>	Web pages.

# How to start an instance of SQL Server automatically (Enterprise Manager)

To start an instance of SQL Server automatically

- 1. Right-click a server, and then click **Properties**.
- 2. Click the **General** tab.
- 3. Under **Autostart policies when the operating system starts**, select the **Autostart SQL Server** check box.

#### See Also

Starting SQL Server Automatically

How to shut off automatic startup of SQL Server (Enterprise Manager)

# How to shut off automatic startup of SQL Server (Enterprise Manager)

#### To shut off automatic startup of SQL Server

- 1. Right-click a server, and then click **Properties.**
- 2. Click the **General** tab.
- 3. Under Autostart policies when the operating system starts, clear the Autostart SQL Server check box.

#### See Also

Starting SQL Server Automatically

How to start SQL Server automatically

#### How to start SQL Server (Enterprise Manager)

#### **To start SQL Server**

• Right-click a server, and then click **Start**.

The green arrow on the icon beside the server name indicates that the server started successfully.

#### See Also

Starting SQL Server Manually Stopping SQL Server Pausing and Resuming SQL Server

# How to stop SQL Server or SQL Server Agent (Enterprise Manager)

**Note** Before stopping an instance of Microsoft® SQL Server<sup>™</sup>, you should pause SQL Server and stop SQL Server Agent to ensure the most orderly shutdown. You can stop both by using SQL Server Service Manager.

#### To stop SQL Server or SQL Server Agent (Enterprise Manager)

- 1. Right-click a server, and then click **Pause**.
- 2. Optionally, send a message informing connected users that the server will be shutting down. After an appropriate interval, proceed to Step 3.
- 3. Right-click SQL Server Agent, and then click **Stop**.
- 4. Right-click the server, and then click **Stop**.

**Note** Stopping an instance of SQL Server by using SQL Server Enterprise Manager or the **net stop mssqlserver** command causes SQL Server to perform a checkpoint in all databases. Then a SHUTDOWN WITH NOWAIT is done to flush all committed data from the data cache and to stop the server immediately.

#### To stop SQL Server or SQL Server Agent (Service Manager)

1. If the service is a remote service, in the **Server** box, enter the name of the remote server. If it is a local server, the Server box will be filled in.

This connects you to the remote server and populates the **Services** box with the names of the SQL Server services registered on the remote computer.

- 2. In the **Services** dialog box, click **SQL Server** or **SQL Server Agent**.
- 3. Click **Pause**.

If you are stopping SQL Server Agent, proceed to Step 4. Otherwise, send a message informing connected users that the server will be shutting down. After an appropriate interval, proceed to Step 4.

4. Click **Stop**.

#### See Also

Stopping SQL Server

# How to start the default instance of SQL Server (Service Manager)

#### To start the default instance of SQL Server

1. In the **Services** box, click **SQL Server**.

If the service is a remote service, type the name of the remote server in the **Server** box.

2. Click **Start/Continue**.

#### See Also

Starting SQL Server Manually

# How to start a clustered instance of SQL Server (Service Manager)

#### To start a clustered instance of SQL Server

- 1. Type the name of the virtual SQL Server in the **Server** box. If it is a default instance, you only need to specify the virtual server name. If it is a named instance, you must enter VIRTUALSERVER\Instance.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Start/Continue**.

#### See Also

Starting SQL Server Manually

#### How to start a named instance of SQL Server (Service Manager)

#### To start a named instance of SQL Server

- 1. In the **Server** box, select the name of the server and the named instance of Microsoft® SQL Server<sup>™</sup> 2000, or type the name of the remote server.
- 2. In the **Services** box, click **SQL Server**, and then click **Start/Continue**.

#### How to start the default instance of SQL Server (Windows)

#### To start the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**, and then click **Start**.

#### How to start a named instance of SQL Server (Windows)

#### To start a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to start, and then click **Start**.

# How to start the default instance of SQL Server (Command Prompt)

To start the default instance of SQL Server from a command prompt

• From a command prompt, enter:

#### sqlservr.exe -c

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>TM</sup> you want to start) in the command window before starting sqlservr.exe.

#### See Also

Starting SQL Server Manually

#### How to start a named instance of SQL Server (Command Prompt)

#### To start a named instance of SQL Server from a command prompt

• From a command prompt, enter this command:

#### sqlservr.exe -c -s {instancename}

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to start) in the command window before starting sqlservr.exe. For example, if Instance1 uses \mssql\$Instance1 to store its binaries, you must be in the \mssql\$Instance1\binn directory to start sqlservr.exe.

## How to start the default instance of SQL Server in single-user mode (Command Prompt)

To start the default instance of SQL Server in single-user mode from a command prompt

• From a command prompt, enter:

#### sqlservr.exe -c -m

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>TM</sup> you want to start) in the command window before starting sqlservr.exe.

#### See Also

Starting SQL Server in Single-User Mode

**Using Startup Options** 

## How to start a named instance of SQL Server in single-user mode (Command Prompt)

To start a named instance of SQL Server in single-user mode from a command prompt

• From a command prompt, enter:

```
sqlservr.exe -c - m -s {instancename}
```

**Note** You must switch to the appropriate directory (for the instance of Microsoft® SQL Server<sup>TM</sup> 2000 you want to start) in the command window before starting sqlservr.exe.

## How to start the default instance of SQL Server with minimal configuration (Command Prompt)

To start the default instance of SQL Server with minimal configuration

• From a command prompt, enter the following command to start the default instance of Microsoft® SQL Server<sup>™</sup> as a service:

sqlservr -c -f

**Note** You must switch to the appropriate directory (for the instance of SQL Server you want to start) in the command window before starting sqlservr.exe.

#### See Also

Starting SQL Server Manually

## How to start a named instance of SQL Server with minimal configuration (Command Prompt)

To start a named instance of SQL Server with minimal configuration

• From a command prompt, enter the following command to start a named instance of Microsoft® SQL Server<sup>™</sup> 2000 as a service:

```
sqlservr -c -f -s {instancename}
```

**Note** You must switch to the appropriate directory (for the instance of SQL Server you want to start) in the command window before starting sqlservr.exe.

# How to pause and resume the default instance of SQL Server (Service Manager)

To pause and resume the default instance of SQL Server

1. In the **Services** box, click **SQL Server**.

If the service is a remote service, type the name of the remote server.

2. Click **Pause**, and then click **Start/Continue**.

### See Also

Pausing and Resuming SQL Server

### How to stop a clustered instance of SQL Server (Service Manager)

#### To stop a clustered instance of SQL Server

- 1. Type the name of the virtual Microsoft® SQL Server<sup>™</sup> in the **Server** box. If it is a default instance, you only need to specify the virtual server name. If it is a named instance, you must enter VIRTUALSERVER\Instance.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Stop**. This pauses the cluster resource, and then stops the **SQL Server** service, which does not cause a failover of SQL Server.

#### See Also

**Stopping SQL Server** 

## How to pause and resume a named instance of SQL Server (Service Manager)

#### To pause and resume a named instance of SQL Server

- 1. In the **Server** box, select the name of the server and the named instance of Microsoft® SQL Server<sup>™</sup> 2000, or type the name of the remote server.
- 2. In the **Services** box, click **SQL Server**.
- 3. Click **Pause**, and then click **Start/Continue**.

# How to pause and resume the default instance of SQL Server (Windows)

To pause and resume the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**.
- 3. Click **Pause** or **Continue**.

# How to pause and resume a named instance of SQL Server (Windows)

To pause and resume a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to pause.
- 3. Click **Pause** or **Continue**.

## How to pause and resume the default instance of SQL Server (Command Prompt)

#### To pause and resume the default instance of SQL Server

• From a command prompt, enter either:

#### net pause mssqlserver

-or-

#### net continue mssqlserver

An instance of Microsoft<sup>®</sup> SQL Server<sup>™</sup> can be paused or resumed only if it was started as a Microsoft Windows NT<sup>®</sup> 4.0 or Windows<sup>®</sup> 2000 service.

## See Also

Pausing and Resuming SQL Server

## How to pause and resume a named instance of SQL Server (Command Prompt)

To pause and resume a named instance of SQL Server

• From a command prompt, enter either:

net pause mssql\$instancename

-or-

net continue mssql\$instancename

## How to broadcast a shutdown message (Command Prompt)

#### To broadcast a shutdown message

• From a command prompt, enter:

**net send** /**users** "message"

For example:

net send /users "SQL Server is going down in 20 minutes. Disconnect within 15 minutes."

**Note** The shutdown message can be broadcast only if an instance of Microsoft® SQL Server<sup>™</sup> is running on Microsoft Windows NT® 4.0 or Windows® 2000. The **users** option specifies that the message be sent to all users connected to the server. For information about other **net send** options, see the Windows NT 4.0 and Windows 2000 documentation.

### See Also

Stopping SQL Server

## How to stop the default instance of SQL Server (Windows)

#### To stop the default instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click **MSSQLSERVER**, and then click **Stop**.

## How to stop a named instance of SQL Server (Windows)

#### To stop a named instance of SQL Server

- 1. In **Control Panel**, double-click **Services**.
- 2. In the **Services** dialog box, click the named instance of Microsoft® SQL Server<sup>™</sup> 2000 you want to stop, and then click **Stop**.

# How to stop the default instance of SQL Server (Command Prompt)

### To stop the default instance of SQL Server

• From a command prompt, enter:

#### net stop mssqlserver

**Note** Stopping a default instance of Microsoft® SQL Server<sup>™</sup> using SQL Server Enterprise Manager or the **net stop mssqlserver** command causes SQL Server to perform a checkpoint in all databases. Then a SHUTDOWN WITH NOWAIT is done to flush all committed data from the data cache and to stop the server immediately. Stopping a default instance of SQL Server from the command prompt works only if you are running Microsoft Windows NT® 4.0 or Windows® 2000.

### See Also

Stopping SQL Server

## How to stop a named instance of SQL Server (Command Prompt)

#### To stop a named instance of SQL Server

• From a command prompt, enter:

#### net stop mssql\$instancename

**Note** Stopping a named instance of Microsoft® SQL Server<sup>™</sup> 2000 using SQL Server Enterprise Manager or the **net stop mssql\$instancename** command causes SQL Server to perform a checkpoint in all databases. Then a SHUTDOWN WITH NOWAIT is done to flush all committed data from the data cache and to stop the server immediately. Stopping a named instance of SQL Server 2000 from the command prompt works only if you are running Microsoft Windows NT® 4.0 or Windows® 2000.

# How to log in to the default instance of SQL Server (Command Prompt)

To log in to the default instance of SQL Server

• From a command prompt, enter either:

osql /U [login\_id] /P [password] /S [servername]

-or-

isql/U [login\_id]/P [password] /S [servername]

## See Also

osql Utility

# How to log in to a named instance of SQL Server (Command Prompt)

To log in to a named instance of SQL Server

• From a command prompt, enter either:

osql / U login\_id /P password /S servername\instancename

-or-

isql/U login\_id/P password /S servername\instancename

## How to change the default service (Service Manager)

#### To change the default service

- 1. Right-click SQL Server Service Manager, and then click **Options**.
- 2. In the **Default Service** box, select the new default service to view through SQL Server Service Manager. When you restart the computer, the service that appears is the new default. For example, if you change the default service to SQLServerAgent service and then shut down the computer, the next time you start it, SQLServerAgent service will be displayed in Service Control Manager. You can only change the default service for the local machine.

## How to create a new failover cluster (Setup)

**IMPORTANT** Before you create a Microsoft® SQL Server<sup>™</sup> 2000 failover cluster, you must configure Microsoft Cluster Service (MSCS) and use Cluster Administrator in Microsoft Windows NT® 4.0 or Windows® 2000 to create at least one cluster disk resource. Note the location of the cluster drive in the Cluster Administrator before you run SQL Server Setup because you need this information to create a new failover cluster.

#### To create a new failover cluster

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and enter a virtual server name. If Setup detects that you are running MSCS, it will default to **Virtual Server**. Click **Next**.
- 3. On the **User Information** screen, enter the user name and company. Click **Next**.
- 4. On the **Software License Agreement** screen, click **Yes**.
- 5. On the **Failover Clustering** screen, enter one IP address for each network configured for client access. That is, enter one IP address for each network on which the virtual server will be available to clients on a public (or mixed) network. Select the network for which you want to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by MSCS. Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

6. On the **Cluster Disk Selection** screen, select the cluster disk group

where the data files will be placed by default. Click **Next**.

- 7. On the **Cluster Management** screen, review the cluster definition provided by SQL Server 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the cluster definition for the virtual server you are creating. Click **Next**.
- 8. On the **Remote Information** screen, enter login credentials for the remote cluster node. The login credentials must have administrator privileges on the remote node(s) of the cluster. Click **Next**.
- 9. On the **Instance Name** screen, choose a default instance or specify a named instance. To specify a named instance, clear the **Default** check box, and then enter the name for the named instance. Click **Next**.

**IMPORTANT** You cannot name an instance DEFAULT or MSSQLSERVER. For more information about naming instances of SQL Server 2000, see <u>Working with Named and Multiple Instances of</u> <u>SQL Server 2000</u>. Names must follow rules for SQL Server identifiers. For more information about naming conventions for identifiers, see <u>Using Identifiers</u>.

10. On the **Setup Type** screen, select the type of installation to install. The Setup program automatically defaults to the first available cluster disk resource from the group you previously selected.

However, if you need to specify a different clustered drive resource, under **Data Files**, click **Browse** and then specify a path on a clustered drive resource. You will be required to select a clustered drive resource that is owned by the node on which you are running the Setup program. The drive also must be a member of the cluster group you previously selected. Click **Next**.

11. On the **Services Accounts** screen, select the service account(s) that you want to run in the failover cluster. Click **Next**.

- 12. In the Authentication Mode dialog box, choose the authentication mode to use. If you change the selection from Windows Authentication Mode to Mixed Mode (Windows Authentication and SQL Server Authentication), you need to enter and confirm a password for the sa login.
- 13. On the **Start Copying Files** screen, click **Next**.
- 14. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from the Setup program when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

## How to install a one-node failover cluster (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and enter a virtual server name. If SQL Server Setup detects that you are running Microsoft® Cluster Service (MSCS), it will default to **Virtual Server**. Click **Next**.
- 3. On the **User Information** screen, enter the user name and company. Click **Next**.
- 4. On the **Software License Agreement** screen, click **Yes**.
- 5. On the **Failover Clustering** screen, enter one IP address per installed network for the virtual server. Select the network for which you wish to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by MSCS. Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

- 6. On the **Cluster Disk Selection** screen, select the cluster disk group where the data files will be placed by default. Click **Next**.
- 7. On the **Cluster Management** screen, review the failover cluster definition provided by Microsoft SQL Server<sup>™</sup> 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the cluster definition for the virtual server you are creating. Click **Next**.

- 8. On the **Remote Information** screen, enter login credentials that have administrator privileges on the remote node of the cluster. Click **Next**.
- 9. On the **Instance Name** screen, choose a default instance or specify a named instance. To specify a named instance, clear the **Default** check box, and then enter the name. Click **Next**.

**IMPORTANT** You cannot name an instance DEFAULT or MSSQLSERVER. The name must follow the rules for SQL Server identifiers. For more information about naming conventions for identifiers, see <u>Using Identifiers</u>.

- 10. On the **Setup Type** screen, select the type of installation to install. Setup will automatically default to the first available clustered disk resource from the group you previously selected. However, if you need to specify a different clustered drive resource, under **Data Files**, click the **Browse** button and then specify a path on a clustered drive resource. You will be required to select a clustered drive resource that is owned by the node on which you are running Setup. The drive must also be a member of the cluster group you previously selected. Click **Next**.
- 11. On the **Services Accounts** screen, select the service account(s) that you want to run in the failover cluster. Click **Next**.
- 12. In the Authentication Mode dialog box, choose the authentication mode to use. If you change the selection from Windows Authentication Mode to Mixed Mode (Windows Authentication and SQL Server Authentication), you must enter and confirm a password for the sa login.
- 13. On the **Start Copying Files** screen, click **Next**.
- 14. On the **Setup Complete** screen, click **Finish**. If you are instructed to restart the computer, do so now. It is important to read the message

from the Setup program when you are done with installation. Failure to restart any of the specified nodes may cause failures when running the Setup program in the future on any node in the cluster.

### How to add nodes to an existing virtual server (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the virtual server to which you want to add a node. Click **Next**.
- 3. On the **Installation Selection** screen, click **Advanced options**. Click **Next**.
- 4. On the **Advanced Options** screen, click **Maintain a virtual server for failover clustering**. Click **Next**.
- 5. On the **Failover Clustering** screen, click **Next**.

You do not need to enter an IP address.

6. On the **Cluster Management** screen, select the node and click **Add**.

If the node is listed as unavailable, you must modify the disk resources in the cluster group of the virtual server so the disk is available for the node you want to add to the Microsoft® SQL Server<sup>™</sup> configuration. Click **Next**.

- 7. On the **Remote Information** screen, enter login credentials for the remote cluster node that has administrator privileges on the remote node of the cluster. Click **Next**.
- 8. On the **Setup Complete** screen, click **Finish**.

## How to remove a node from an existing failover cluster (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the name of the server from which to remove the node. Click **Next**.
- 3. You may see an error message saying that one (or more) of the nodes of the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 cluster are unavailable. This may be because the node(s) you are attempting to remove is damaged. The node(s) still can be removed. Click **OK**.
- 4. On the **Installation Selection** screen, click **Advanced Options**. Click **Next**.
- 5. On the **Advanced Options** screen, click **Maintain a virtual server for failover clustering**. Click **Next**.
- 6. On the **Failover Clustering** screen, click **Next**.

You do not need to modify any IP address(es).

- 7. On the **Cluster Management** screen, select the node and click **Remove**. Click **Next**.
- 8. On the **Remote Information** screen, enter login credentials for the remote cluster node that has administrator privileges on the remote node(s) of the cluster. Click **Next**.
- 9. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from SQL Server Setup when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

### How to remove a failover clustered instance (Setup)

- 1. On the **Welcome** screen of the **Microsoft SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Virtual Server** and specify the name of the server from which to remove a clustered instance. Click **Next**.
- 3. On the **Installation Selection** screen, click **Upgrade, remove, or add components to an existing instance of SQL Server**.
- 4. On the **Instance Name** screen, for a default instance, click **Default**. For a named instance, specify the name of the instance to remove. Click **Next**.
- 5. On the **Existing Installation** screen, click **Uninstall your existing installation**. Click **Next**.
- 6. On **the Remote Information** screen, specify the password that is a valid administrator password on all nodes in the cluster. Click **Next**.
- 7. In the **Setup** message "Successfully uninstalled the instance . . . ", click **OK**.
- 8. On the **Setup Complete** screen, click **Finish**.

If you are instructed to restart the computer, do so now. It is important to read the message from SQL Server Setup when you are done with installation. Failure to restart any of the specified nodes may cause failures when you run the Setup program in the future on any node in the failover cluster.

## How to recover from failover cluster failure in Scenario 1

In this scenario, failure is caused by hardware failure in Node 1 of a two-node cluster. This hardware failure could be caused, for example, by the failure of a small computer system interface (SCSI) card or the operating system.

- 1. After Node 1 fails, the Microsoft® SQL Server<sup>™</sup> 2000 failover cluster fails over to Node 2.
- 2. Run SQL Server Setup and remove Node 1. For more information, see <u>How to remove a failover clustered instance</u>.
- 3. Evict Node 1 from Microsoft Cluster Service (MSCS). To evict a node from MSCS, from Node 2, right-click on the node to remove, and then click **Evict Node**.
- 4. Install new hardware to replace the failed hardware in Node 1.
- 5. Install the operating system. For more information about which operating system to install and specific instructions on how to do this, see <u>Before Installing Failover Clustering</u>.
- 6. Install MSCS and join the existing cluster. For more information, see <u>Before Installing Failover Clustering</u>.
- 7. Run the Setup program on Node 2 and add Node 1 back to the failover cluster. For more information, see <u>How to add nodes to an existing virtual server (Setup)</u>.

## How to recover from failover cluster failure in Scenario 2

In Scenario 2, failure is caused by Node 1 being down or offline but not irretrievably broken. This could be caused, for example, by an operating system failure.

- 1. After Node 1 fails, the Microsoft® SQL Server<sup>™</sup> 2000 failover cluster fails over to Node 2.
- 2. Run SQL Server Setup and remove Node 1. For more information, see <u>How to remove a failover clustered instance</u>.
- 3. Resolve the problem with Node 1.
- 4. Ensure that the Microsoft Cluster Service (MSCS) cluster is working and all nodes are online.
- 5. Run the Setup program on Node 2 and add Node 1 back to the failover cluster. For more information, see <u>How to add nodes to an existing virtual server (Setup)</u>.

# How to upgrade from a SQL Server 6.5 active/passive failover cluster (Setup)

#### To upgrade from a SQL Server 6.5 active/passive failover cluster

- 1. Uncluster Microsoft<sup>®</sup> SQL Server<sup>™</sup> version 6.5.
- 2. Install a default instance of SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. On all nodes of the cluster, this drive must have at least 300 megabytes (MB) of available space.

- 3. Run the SQL Server Upgrade Wizard to migrate your data into SQL Server 2000.
- 4. Uninstall SQL Server 6.5.
- 5. Run SQL Server Setup to upgrade your default instance of SQL Server 2000 to a SQL Server 2000 failover cluster.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

#### See Also

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a</u> <u>direct pipeline (SQL Server Upgrade Wizard)</u>

<u>How to perform a SQL Server version 6.5 to SQL Server 2000 upgrade using a tape drive (SQL Server Upgrade Wizard)</u>

# How to upgrade from a SQL Server 6.5 active/active failover cluster (Setup)

**Note** To upgrade from a Microsoft® SQL Server<sup>™</sup> 6.5 active/active failover cluster (or any configuration where SQL Server exists on the second node), you must first convert one side of the failover cluster to a named instance of SQL Server 2000.

#### To upgrade from a SQL Server 6.5 active/active failover cluster

- 1. On Node 1, uncluster SQL Server 6.5. On Node 2, uncluster SQL Server 6.5.
- 2. On Node 1, install a default (non-clustered) instance of SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

- 3. On Node 1, run the SQL Server 2000 Upgrade Wizard to migrate your data into SQL Server 2000.
- 4. On Node 1, uninstall the instance of SQL Server 6.5.
- 5. On Node1, install a named, clustered instance of SQL Server 2000.
- 6. Run the Copy Database Wizard (CDW.exe) to migrate your SQL Server data (originally from SQL Server 6.5) to a named instance in a SQL Server 2000 failover cluster. For more information about the Copy Database Wizard, see <u>Using the Copy Database Wizard</u> or <u>How</u> to upgrade databases online using the Copy Database Wizard (Enterprise Manager).

- 7. On Node 1, uninstall the default instance of SQL Server 2000.
- 8. On Node 2, install a default instance of SQL Server 2000.
- 9. Run the SQL Server 2000 Upgrade Wizard to migrate your data into SQL Server 2000.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. On all nodes of the cluster, this drive must have at least 300 megabytes (MB) of available space.

- 10. On Node 2, uninstall the instance of SQL Server 6.5.
- 11. On Node 2, upgrade the default instance of SQL Server to a clustered instance.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

## How to upgrade from a SQL Server 7.0 active/active failover cluster (Setup)

**Note** To upgrade from a Microsoft® SQL Server<sup>™</sup> version 7.0 active/active failover cluster (or any configuration where SQL Server exists on the second node), you must first convert one side of the failover cluster to a named instance of SQL Server 2000.

#### To upgrade from a SQL Server 7.0 active/active failover cluster

- 1. On Node 1, uncluster SQL Server version 7.0. Reboot Node 1.
- 2. On Node 2, uncluster SQL Server 7.0. Reboot Node 2.
- 3. On Node 1, install a clustered, named instance of SQL Server 2000 as a virtual server. This is not an upgrade process, but a side-by-side installation of SQL Server 7.0 and SQL Server 2000. Do not install the data to the same location/disk as Node 2. If you do, when you attempt to upgrade Node 2 from a SQL Server 7.0 to a SQL Server 2000 installation, Setup will fail.
- 4. On Node 1, run the Copy Database Wizard (CDW.exe) to move all databases and related information from the SQL Server 7.0 installation into the clustered, named instance of SQL Server 2000. For more information about the Copy Database Wizard, see <u>Using the Copy</u> Database Wizard or How to upgrade databases online using the Copy Database Wizard (Enterprise Manager).
- 5. On Node 1, uninstall SQL Server 7.0.
- 6. On Node 2, upgrade SQL Server 7.0 to SQL Server 2000 as the default instance.

You must install the binaries to a local drive and use a cluster disk for

the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

7. On Node 2, upgrade the default instance of SQL Server 2000 to a clustered instance.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

**Note** Optionally, you could create two named instances of SQL Server 2000 and use the Copy Database Wizard to upgrade both SQL Server 7.0 installations to a clustered, named instance of SQL Server 2000. This will provide better consistency, because all references to clustered installations of SQL Server 2000 will be in the form VirtualServer\Instance, rather than sometimes being just the servername, and sometimes both the servername and instancename.

## How to upgrade from a SQL Server 7.0 active/passive failover cluster (Setup)

#### To upgrade from a SQL Server 7.0 active/passive failover cluster

- 1. On Node 1, uncluster Microsoft® SQL Server™ version 7.0. Reboot Node 1.
- 2. On Node 1, upgrade SQL Server 7.0 to SQL Server 2000 as the default instance.

You must install the binaries to a local drive and use a cluster disk for the data. This local drive is a path, which is a non-clustered disk valid on all nodes of the cluster. This drive on all nodes of the cluster must have at least 300 megabytes (MB) of available space.

3. On Node 1, upgrade the default instance of SQL Server 2000 to a clustered instance of SQL Server 2000.

For more information, see <u>How to upgrade from a default instance to a</u> <u>default clustered instance of SQL Server 2000 (SQL Server Setup)</u>.

# How to upgrade from a default instance to a default clustered instance of SQL Server 2000 (Setup)

**Note** This upgrade is from a default instance (a local installation where the data is on a local disk) to a clustered instance of Microsoft® SQL Server<sup>™</sup> 2000. Use this upgrade step if you want to have a default virtual server.

# To upgrade from a default instance to a default clustered instance of SQL Server 2000

- 1. On the **Welcome** screen of the **SQL Server Installation Wizard**, click **Next**.
- 2. On the **Computer Name** screen, click **Local Computer**. The computer you want to change from a default to a clustered instance should be displayed. You must be on the local computer to upgrade from a default to a clustered instance. Click **Next**.
- 3. On the **Installation Selection** screen, click **Upgrade, remove, or add components to an existing instance of SQL Server**. Click **Next**.
- 4. On the **Existing Installation** screen, click **Upgrade your existing installation to a clustered installation**. Click **Next**.
- 5. On the **Virtual Server Name** screen, enter a name for your virtual server. Click **Next**.
- 6. On the **Failover Clustering** screen, enter one IP address for each network configured for client access. That is, enter one IP address for each network on which the virtual server will be available to clients on a public (or mixed) network. Select the network for which you want to enter an IP address, and then enter the IP address. Click **Add**.

The IP address and the subnet are displayed. The subnet is supplied by

Microsoft Cluster Service (MSCS). Continue to enter IP addresses for each installed network until you have populated all desired networks with an IP address. Click **Next**.

- 6. On the **Cluster Management** screen, review the failover cluster definition provided by SQL Server 2000. By default, all available nodes are selected. Remove any nodes that will not be part of the failover cluster definition for the virtual server you are creating. Click **Next**.
- 7. On the **Remote Information** screen, enter login credentials for the remote cluster node. The login credentials must have administrator privileges on the remote node(s) of the cluster. Click **Next**.
- 8. On the **Services Accounts** screen, select the service account(s) for the SQL Server services under which you want the failover cluster to run. Click **Next**.
- 9. On the **Setup Complete** screen, click **Finish**. If you need to restart the remote nodes in the failover cluster, you will be instructed to do so in the **Setup Complete** screen.

# How to upgrade from a local default instance to a clustered, named instance of SQL Server 2000 (Setup)

To upgrade from a local default instance to a named clustered instance of SQL Server 2000

- 1. Install a clustered, named instance of Microsoft® SQL Server<sup>™</sup> 2000.
- 2. Run the Copy Database Wizard (CDW.exe) to move all databases and related information into the clustered, named instance of SQL Server 2000. For more information about the Copy Database Wizard, see Using the Copy Database Wizard or How to upgrade databases online using the Copy Database Wizard (Enterprise Manager).
- 3. Optionally, you can uninstall the default instance of SQL Server 2000.

# **Backing Up and Restoring Databases**

The backup and restore component of Microsoft® SQL Server<sup>™</sup> provides an important safeguard for protecting critical data stored in SQL Server databases. Understanding how to create and restore database, differential database, transaction log, and file and filegroup backups helps you implement this important safeguard.

# How to create a logical disk backup device (Enterprise Manager)

#### To create a logical disk backup device

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **Backup**, and then click **New Backup Device**.
- 3. In the **Name** box, type a name for the named backup device.
- 4. Click **File name**, and then do one of the following:
  - Type the name of the file used by the disk backup device.
  - Click the browse (...) button to display the **Backup Device Location** dialog box, and then select the file on the local computer used by the disk backup device.

## See Also

**Backup Devices** 

# How to create a logical tape backup device (Enterprise Manager)

### To create a logical tape backup device

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **Backup**, and then click **New Backup Device**.
- 3. In the **Name** box, type a name for the named backup device.
- 4. Click **Tape drive name**, and then click the tape device to use as the tape backup device.

**Note** If no tape devices are listed, then no tape devices can be detected on the local computer. For more information about how to set up tape devices, see the Microsoft® Windows NT® 4.0 and Windows® 2000 documentation.

## See Also

**Backup Devices** 

# How to delete a logical backup device (Enterprise Manager) To delete a logical backup device

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then click **Backup**.
- 3. In the details pane, right-click the named backup device to delete.
- 4. Click **Delete**, and then confirm the deletion.

# See Also

**Backup Devices** 

sp\_dropdevice

# How to create a database backup (Enterprise Manager)

#### To create a database backup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Backup Database**.
- 3. In the **Name** box, type the backup set name. Optionally, in **Description**, type a description of the backup set.
- 4. Under **Backup**, click **Database complete**.
- 5. Under **Destination**, click **Tape** or **Disk**, and then specify a backup destination.

If no backup destinations appear, click **Add** to add an existing destination or to create a new one.

- 6. Under **Overwrite**, do one of the following:
  - Click **Append to media** to append the backup to any existing backups on the backup device.
  - Click **Overwrite existing media** to overwrite any existing backups on the backup device.
- 7. Optionally, select the **Schedule** check box to schedule the backup operation for later or periodic execution.
- 8. Optionally, click the **Options** tab and do one or more of the following:
  - Select the **Verify backup upon completion** check box to cause the backup to be verified when backed up.

- Select the **Eject tape after backup** check box to cause the tape to be ejected when the backup operation has completed. Available only with tape devices.
- Select the **Check media set name and backup set expiration** check box to cause the backup media to be checked to prevent accidental overwrites. In **Media set name**, type the name of the media to be used for the backup operation. Leave blank when specifying only the backup set expiration.
- 9. If it is the first use of the backup media, or you want to change an existing media label, under Media set labels, select the Initialize and label media check box and type the media set name and media set description. The media can be initialized and labeled only when overwriting the media.

## See Also

Appending Backup Sets

Backing Up the master Database

Backing Up the model, msdb, and distribution Databases

**Copying Databases** 

**Transaction Log Backups** 

**Database Backups** 

**Differential Database Backups** 

Deleting a Database

**Overwriting Backup Media** 

**Reducing Recovery Time** 

**Initializing Backup Media** 

Verifying Backups

# How to start the Create Database Backup Wizard (Enterprise Manager)

## To start the Create Database Backup Wizard

- 1. Expand a server group, and then expand a server.
- 2. On the **Tools** menu, click **Wizards**.
- 3. In the **Select Wizard** dialog box, expand **Management**.
- 4. Double-click **Backup Wizard**.
- 5. Complete the steps in the wizard.

## See Also

**Database Backups** 

# How to restore a database backup (Enterprise Manager)

**Note** If you are restoring a database backup that does not have any backup set information listed in the backup history stored in the **msdb** database, such as a database backup created on another server, see <u>How to restore a backup from a backupdevice</u>.

#### To restore a database backup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore, if different from the default. To restore the database with a new name, type the new name of the database.
- 4. Click **Database**.
- 5. In the **First backup to restore** list, click the backup set to restore.
- 6. In the **Restore** list, click the database backup to restore.
- 7. Optionally, click the **Options** tab and do the following:
  - In **Restore as**, type the new name or location for each database file comprising the database backup.

**Note** Specifying a new name for the database determines automatically the new names for the database files restored from the database backup.

• Click Leave database operational. No additional transaction logs can be

**restored** if no further transaction log or differential database backups are to be applied.

• Click **Leave database nonoperational, but able to restore additional transaction logs** if another transaction log or differential database backup is to be applied.

## See Also

Transaction Log Backups Database Backups Differential Database Backups Identifying the Backup Set to Restore Rebuilding the master Database Restoring a Database to a Prior State Restoring the master Database from a Current Backup Restoring the model, msdb, and distribution Databases Reducing Recovery Time

# How to restore a backup from a backup device (Enterprise Manager)

#### To restore a backup from a backup device

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore if different from the default. To restore the database with a new name, type the new name of the database.

**Note** Specifying a new name for the database determines automatically the new names for the database files restored from the database backup.

- 4. Click **From device**, and then click **Select devices**.
- 5. Under **Restore from**, click **Tape** or **Disk**, and then select a device from which to restore.

If no devices appear, click **Add** to add an existing backup device or to create a new one. In the **Restore Database** dialog box, click **View Contents** and select the backup set to restore.

**Note** This option scans the backup set for the backup content information and can be time consuming, especially when using tape devices. If you already know the backup set to restore, type the backup set number in **Backup number** instead.

- 6. Under **Restore backup set**, do one of the following:
  - Click **Database complete** to restore a database backup.

- Click **Database differential** to restore a differential database backup.
- Click **Transaction log** to apply a transaction log backup.
- Click **File or filegroup** to restore a file or filegroup backup. Specify the name of the file or filegroup.
- 7. Optionally, click the **Options** tab, and then do one of the following:
  - Click Leave database operational. No additional transaction logs can be restored if no further transaction log backups are to be applied.
  - Click Leave database nonoperational, but able to restore additional transaction logs if another transaction log backup is to be applied.

## See Also

Transaction Log Backups

Database Backups

**Differential Database Backups** 

Identifying the Backup Set to Restore

Rebuilding the master Database

Restoring a Database to a Prior State

Restoring the master Database from a Current Backup

Restoring the model, msdb, and distribution Databases

Using Differential Database Backups with Transaction Log Backups

# How to create a transaction log backup (Enterprise Manager)

#### To create a transaction log backup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Backup Database**.
- 3. In the **Name** box, type the backup set name. Optionally, in **Description**, type a description of the backup set.
- 4. Under **Backup**, click **Transaction log**.

**Note** If the **Transaction Log** option is unavailable, ensure that the recovery model is set to **Full** or **Bulk-Logged**. For more information, see <u>Using Recovery Models</u>.

5. Under **Destination**, click **Tape** or **Disk**, and then specify a backup destination.

If no backup destinations appear, click **Add** to add an existing backup device or to create a new one.

- 6. Under **Overwrite**, do one of the following:
  - Click **Append to media** to append the backup to any existing backups on the backup device.
  - Click **Overwrite existing media** to overwrite any existing backups on the backup device.
- 7. Optionally, select the **Schedule** check box to schedule the backup operation for later or periodic execution.

- 8. Optionally, click the **Options** tab, and then do one of the following:
  - Select the **Verify backup upon completion** check box to cause the backup to be verified when backed up.
  - Select the **Eject tape after backup** check box to cause the tape to be ejected when the backup operation has completed. Available only with tape devices.
  - Select the **Remove inactive entries from transaction log** check box to cause the inactive portion of the transaction log to be truncated, allowing Microsoft® SQL Server<sup>™</sup>
  - to reuse this truncated, unused space.
  - Select the **Check media set name and backup set expiration** check box to cause the backup media to be checked to prevent accidental overwrites. In the **Media set name** box, type the name of the media to be used for the backup operation. Leave blank when specifying only the backup set expiration.
- 9. If it is the first use of the backup media or you want to change an existing media label, under Media set labels, select the Initialize and label media check box and type the media set name and media set description. The media can only be initialized and labeled when it is being overwritten.

# See Also

<u>Transaction Log Backups</u> <u>Using File Backups</u> <u>Restoring a Database to a Prior State</u> <u>Reducing Recovery Time</u>

# How to apply a transaction log backup (Enterprise Manager)

**Note** If you are restoring a transaction log backup that does not have any backup set information listed in the backup history stored in the **msdb** database, such as a transaction log backup created on another server, see <u>How to restore a</u> <u>backup from a backup device</u>.

#### To apply a transaction log backup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore, if different from the default.
- 4. Click **Database**.
- 5. In the **First backup to restore** list, click the backup set to restore.
- 6. In the **Restore** list, click the transaction log backup to restore.
- 7. Optionally, click the **Options** tab and , and then do one of the following:
  - Click Leave database operational. No additional transaction logs can be restored if no further transaction log or differential database backups are to be applied.
  - Click **Leave database nonoperational, but able to restore additional transaction logs** if another transaction log or differential database backup is to be applied.

# See Also

Transaction Log Backups Using File Backups Reducing Recovery Time

# How to create a differential database backup (Enterprise Manager)

### To create a differential database backup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database, point to **All Tasks**, and then click **Backup Database**.
- 3. In the **Name** box, type the backup set name. Optionally, in **Description**, type a description of the backup set.
- 4. Under **Backup**, select **Database differential**.
- 5. Under **Destination**, click **Tape** or **Disk**, and then specify a backup destination.

If no backup destinations appear, click **Add** to add an existing backup device or to create a new one.

- 6. Under **Overwrite**, do one of the following:
  - Click **Append to media** to append the backup to any existing backups on the backup device.
  - Click **Overwrite existing media** to overwrite any existing backups on the backup device.
- 7. Optionally, select the **Schedule** check box to schedule the backup operation for later or periodic execution.
- 8. Optionally, click the **Options** tab, and then do one or more of the following:

- Select the **Verify backup upon completion** check box to cause the backup to be verified when backed up.
- Select the **Eject tape after backup** check box to cause the tape to be ejected when the backup operation has completed. Available only with tape devices.
- Select the **Check media set name and backup set expiration** check box to cause the backup media to be checked to prevent accidental overwrites. In **Media set name**, type the name of the media to be used for the backup operation. Leave blank when specifying only the backup set expiration.
- 9. If it is the first use of the backup media or you want to change an existing media label, under Media set labels, select the Initialize and label media check box and type the media set name and media set description. The media can be initialized and labeled only when overwriting the media.

# See Also

**Differential Database Backups** 

**Reducing Recovery Time** 

# How to restore a differential database backup (Enterprise Manager)

**Note** If you are restoring a differential database backup that does not have any backup set information listed in the backup history stored in the **msdb** database, such as a differential database backup created on another server, see <u>How to</u> <u>restore a backup from a backup device</u>.

#### To restore a differential database backup

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore, if different from the default.
- 4. Click **Database**.
- 5. In the **First backup to restore** list, click the backup set to restore.
- 6. In the **Restore** list, click the differential backup to restore.
- 7. Optionally, click the **Options** tab, and then do one of the following:
  - Click **Leave database operational**. **No additional transaction logs can be restored** if no further transaction log backups are to be applied.
  - Click Leave database nonoperational, but able to restore additional transaction logs if another transaction log backup is to be applied.

# See Also

Differential Database Backups

Reducing Recovery Time

# How to set up, maintain, and bring online a standby server (Enterprise Manager)

Setting up a standby server generally involves creating database backups and periodic transaction log backups at the primary server, and then applying those backups, in sequence, to the standby server. The standby server is left in a read-only state between restore operations. When the standby server must be made available for use, any outstanding transaction log backups from the primary server, including the backup of the active transaction log, are applied to the standby server; then the database is recovered.

#### To create backups on the primary server

How to create a database backup

How to create a transaction log backup

#### To set up and maintain a backup (standby) server

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In **Restore as database**, type or select the name of the database to restore, if different from the default. To restore the database with a new name, type the new name of the database.
- 4. Under **Restore**, click **From device**, and then click **Select devices**.
- 5. Under **Restore from,** click **Tape** or **Disk**, and then select a device from which to restore.

If no devices appear, click **Add** to add an existing backup device or to create a new one. The backup device must reference the backup device files created at the primary server.

6. In the **Restore Database** dialog box, click **View contents**. Select the backup set to restore.

**Note** This option scans the backup set for the backup content information and can be time consuming, especially when using tape devices. If you already know the backup set to restore, type the backup set number in **Backup number** instead.

- 7. Under **Restore backup set**, do one of the following:
  - Click **Database complete** to restore the initial database backup created on the primary server. The initial database backup must be restored before any transaction log backups can be applied.
  - Click **Transaction log** to apply a transaction log backup created on the primary server.
- 8. On the **Options** tab, click **Leave database read-only and able to restore additional transaction logs**, and then in the **Undo** file box, type the name of the undo file that contains the contents of data pages before uncommitted transactions affecting those pages were rolled back.
- 9. Repeat this procedure for each transaction log backup applied to the standby server.

## To bring the standby server online (primary server failed)

1. Back up the active transaction log on the primary server, if possible.

For more information, see <u>How to create a backup of the currently</u> <u>active transaction log</u>.

2. Apply all transaction log backups, including the active transaction log backup created in Step 1, which have not yet been applied to the standby server.

For more information, see <u>How to apply a transaction log backup</u>.

3. Recover the database.

For more information, see <u>How to recover a database without</u> <u>restoring</u>.

# See Also

**Using Standby Servers** 

# How to restore to a point in time (Enterprise Manager)

#### To restore to a point in time

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In **Restore as database**, type or select the name of the database to restore, if different from the default.
- 4. Click **Database**.
- 5. In the **First backup to restore** list, click the backup set to restore.
- 6. In the **Restore** list, select the database backup and one or more transaction logs to restore.
- 7. Click **Point in time restore**, and then type values for **Date** and **Time**.
- 8. Click the **Options** tab, and then click **Leave database operational**. **No additional transaction logs can be restored**.

## See Also

Restoring a Database to a Prior State

# How to view the data and log files in a backup set (Enterprise Manager)

## To view the data and log files in a backup set

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. Click **From device**, and then click **Select devices**.
- 4. Under **Restore from,** click **Tape** or **Disk**, and then select a device from which to restore.

If no backup destinations appear, click **Add** to add an existing backup device or to create a new one. The backup device must reference the backup device files created at the primary server.

5. In the **Restore Database** dialog box, click **View contents**.

## See Also

Verifying Backups

**Viewing Information about Backups** 

# How to view backup and media header information (Enterprise Manager)

## To view backup and media header information

- 1. Expand a server group, and then expand a server.
- 2. Expand Management, and then click **Backup**.
- 3. In the details pane, right-click the named backup device to view, and then click **Properties**.
- 4. Click **View Contents**.

# See Also

<u>Verifying Backups</u> <u>Viewing Information about Backups</u>

# How to back up files and filegroups (Enterprise Manager)

### To back up files and filegroups

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Backup Database**.
- 3. In the **Name** box, type the backup set name. Optionally, in **Description**, type a description of the backup set.
- 4. Under Backup, click **File and filegroup**, and then click the browse (...) button.
- 5. In the **Specify Filegroups and Files** dialog box, select a **Backup** for each filegroup or file you want to back up.

By selecting a filegroup name, all the files within the filegroup are selected automatically.

6. Under **Destination**, click **Tape** or **Disk**, and then specify a backup destination.

If no backup destinations appear, click **Add** to add an existing backup destination or to create a new one.

- 7. Under **Overwrite**, do one of the following:
  - Click **Append to media** to append the backup to any existing backups on the backup device.
  - Click **Overwrite existing media** to overwrite any existing backups on the backup device.
- 8. Optionally, the **Schedule** check box to schedule the backup operation

for later or periodic execution.

- 9. Optionally, click the **Options** tab, and then do one or more of the following:
  - Select the **Verify backup upon completion** check box to cause the backup to be verified when backed up.
  - Select the **Eject tape after backup** check box to cause the tape to be ejected when the backup operation has completed. Available only with tape devices.
  - Select the **Check media set name and backup set expiration** check box to cause the backup media to be checked to prevent accidental overwrites. In **Media set name**, type the name of the media to be used for the backup operation. Leave blank when specifying only the backup set expiration.
- 10. If it is the first use of the backup media or you want to change an existing media label, under **Media set labels**, select the **Initialize and label media** check box and type the media set name and media set description. The media can be initialized and labeled only when it is being overwritten.

## See Also

Using File Backups

# How to restore files and filegroups (Enterprise Manager)

**Note** If you are restoring a file or filegroup backup that does not have any backup set information listed in the backup history stored in the **msdb** database, such as a file or filegroup backup created on another server, see <u>How to restore a backup from a backup device</u>.

#### To restore files and filegroups

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore, if different from the default.
- 4. Click **Filegroups or files**.
- 5. In the **Restore** list, select each file and filegroup to restore.
- 6. Click the **Options** tab, and then do one of the following:
  - Click **Leave database operational**. **No additional transaction logs can be restored** if no further transaction log backups are to be applied.
  - Click Leave database nonoperational, but able to restore additional transaction logs if another transaction log backup is to be applied.

**IMPORTANT** If the files have been modified since the file backup was created, transaction log backups created after the file backup must be applied.

See Also

Using File Backups

# How to restore files and filegroups over existing files (Enterprise Manager)

#### To restore files and filegroups over existing files

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, point to **All Tasks**, and then click **Restore Database**.
- 3. In the **Restore as database** box, type or select the name of the database to restore, if different from the default.
- 4. Click **From device**, and then click **Select devices**.
- 5. Under **Restore from**, click **Tape** or **Disk**, and then select a device from which to restore.

If no backup destinations appear, click **Add** to add an existing backup device or to create a new one. The backup device must reference the backup device files created at the primary server.

6. In the **Restore Database** dialog box, click **View contents**. Select the backup set to restore.

**Note** This option scans the backup set for the backup content information and can be time consuming, especially when using tape devices. If you already know the backup set to restore, type the backup set number in **Backup number** instead.

- 7. Under **Restore backup set**, click **File or filegroup**, and then type the names of the files you want to restore.
- 8. Click the **Options** tab, and then click **Force restore over existing**

#### database.

- 9. Under **Recovery completion state**, do one of the following:
  - Click Leave database operational. No additional transaction logs can be restored if no further transaction log backups are to be applied.
  - Click **Leave database nonoperational, but able to restore additional transaction logs** if another transaction log backup is to be applied.

# See Also

**Copying Databases** 

# How to set the recovery model for a database (Enterprise Manager)

#### To set the recovery model for a database

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, right-click the database, and then click **Properties**.
- 3. Click the **Options** tab.
- 4. In the **Model** list, click a recovery model.

### See Also

**Using Recovery Models** 

## **Managing Servers**

Microsoft® SQL Server<sup>™</sup> server management comprises a wide variety of administration tasks, including:

- Registering servers and assigning passwords.
- Reconfiguring network connectivity.
- Configuring linked servers, which allows you to execute distributed queries and distributed transactions on OLE DB data sources across the enterprise.
- Configuring remote servers, which allows you to use one instance of SQL Server to execute a stored procedure residing on another instance of SQL Server.
- Configuring standby servers.
- Setting server configuration options.
- Managing SQL Server messages.
- Setting the polling intervals.

In most cases, you do not need to reconfigure the server. The default settings for server components, configured during SQL Server Setup, allow you to run an instance of SQL Server immediately after SQL Server is installed. However, server management is necessary in those situations where you want to add new servers, set up special server configurations, change the network connections, or set server configuration options to improve SQL Server performance.

## How to register a server (Enterprise Manager)

#### To register a server

1. Right-click a server or a server group, and then click **New SQL Server Registration**.

Note If you selected the From now on I want to perform this task without using a wizard check box the last time you used the Register Server Wizard, SQL Server Enterprise Manager displays the Registered SQL Server Properties dialog box. Otherwise, the Register Server Wizard is started.

- 2. In the **Server** box, type the server name.
- 3. To specify the connection between SQL Server Enterprise Manager (as a client) and the server running the instance of Microsoft® SQL Server<sup>™</sup> being registered, do one of the following:
  - Click Use Windows Authentication

-or-

• Click Use SQL Server Authentication

With this type of connection, you must provide a login name and password. Select the **Always prompt for login name and password** check box to always prompt user for login name and password, rather than storing your login id and password in your registry.

4. In the **Server Group** list, click a server group.

**Note** If the group you need does not exist yet, create it by clicking the build (...) button, and then completing the **Server Groups** dialog box.

- 5. Do one or more of the following:
  - Select the **Display SQL Server state in console** check box to

turn on service polling.

- Select the **Show system databases and system objects** check box to show all system databases and objects.
- Select the **Automatically start SQL Server when connecting** check box to start an instance of SQL Server automatically.
- 6. Repeat Steps 1 through 5 to register each server with SQL Server Enterprise Manager.

### See Also

How to create server groups

Managing Servers

**Registering Servers** 

### How to create server groups (Enterprise Manager)

#### To create server groups

- 1. Right-click a server group, and then click **New SQL Server Registration**.
- 2. Under **Options**, click the add (...) button.
- 3. In the **Name** box, enter a unique name for the new group.
- 4. Choose from the following group levels:
  - Top level group
  - Sub-group of

If this option is selected, you need to select the top level group under which you want the new subgroup to be below.

5. Repeat Steps 2 through 4 to create each new server group.

### See Also

How to register a server (Enterprise Manager)

Managing Servers

**<u>Registering Servers</u>** 

## How to change a server's registration (Enterprise Manager)

#### To change a server's registration

- 1. Expand a server group, and then right-click a server.
- 2. Click **Edit SQL Server Registration properties**, and then change the server's registration as appropriate.

### See Also

**Registering Servers** 

## How to remove a registered server running SQL Server (Enterprise Manager)

To remove a registered server running SQL Server

- 1. Expand a server group, and then right-click a server.
- 2. Click **Delete SQL Server Registration**.
- 3. Confirm the deletion.

## How to connect to a registered server running SQL Server (Enterprise Manager)

To connect to a registered server running SQL Server

- 1. Expand a server group, and then right-click a server.
- 2. Click **Connect** or expand the server.

## How to disconnect from a registered server running SQL Server (Enterprise Manager)

To disconnect from a registered server running SQL Server

- 1. Expand a server group, and then right-click a server.
- 2. Click **Disconnect**.

## How to assign the sa password on a newly installed server (Enterprise Manager)

#### To assign the sa password

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click **sa**, and then click **Properties**.
- 4. In the **Password** box, type the new password.

## How to view server properties (Enterprise Manager)

### To view server properties

- 1. Expand a server group, and then right-click a server.
- 2. Click **Properties**.

## How to check and set remote server configuration options (Enterprise Manager)

#### To check and set remote server configuration options

- 1. Expand a server group, and then right-click a server.
- 2. Click **Properties,** and then click the **Connections** tab.
- 3. Under **Remote server connections**, review and, if appropriate, change the values for the following configuration options:
  - Allow other SQL Servers to connect remotely to this SQL Server using RPC.
  - Query time-out (sec, 0 = unlimited)

This option specifies the number of seconds to wait before returning from processing a query. A value of 0 will allow an infinite wait. The default is 0.

• Enforce distributed transactions (MTS).

Changing a configuration option requires that you stop and restart the server. If you changed a configuration option, then proceed to Step 4. If not, skip to Step 6.

- 4. Right-click the server, and then click **Stop**.
- 5. After the server has stopped, right-click the server, and then click **Start**.
- 6. Repeat Steps 1 through 5 on the other server of the remote server pair.

## How to set up a central store for server registration information (Enterprise Manager)

To set up a central store for server registration information on a remote server

- 1. Click the server, and then on the **Tools** menu, click **Options**.
- 2. On the **General** tab, clear the **Read/Store user independent** check box.
- 3. Use SQL Server Enterprise Manager to create server groups and register servers.

#### To set up a central store for server registration information on a local server

- 1. Click the server, and then on the **Tools** menu, click **Options**.
- 2. On the **General** tab, click **Read from remote**.
- 3. In the **Server name** box, type the name of the remote server from which you want to use registration information.

## How to set access to your display of servers and groups (Enterprise Manager)

To set access to your display of servers and groups

- 1. Expand a server group, and then click a server.
- 2. On the **Tools** menu, click **Options**.
- 3. On the **General** tab, click **Read/Store locally**, and then select or clear the **Read/Store user independent** check box.

## How to set the polling interval (Enterprise Manager)

#### To set the polling interval

- 1. Expand a server group, and then click a server.
- 2. On the **Tools** menu, click **Options**.
- 3. On the **General** tab, select the **Poll server to find out state of server and related services** check box.
- 4. In the **Service** list, click the service to poll.
- 5. In the **Poll interval (seconds)** box, type or select the polling interval.

#### See Also

How to set the polling interval (Service Manager)

## How to disable a remote server setup (Enterprise Manager)

#### To disable a remote server setup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Remote Servers** to list the remote servers defined on the selected server.
- 3. Right-click the remote server to disable, and then click **Delete**.

# How to manage or view SQL Server messages (Enterprise Manager)

#### To manage or view SQL Server messages

- 1. Expand a server group, and then right-click a server.
- 2. Point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. To specify search options, do one or more of the following:
  - In the **Message text contains** box, type the text to search for.
  - In the **Error number** box, type the error number to search for.
  - Under **Severity**, select a severity level to search for.
- 4. To specify the messages to include, do one of the following:
  - Select the **Only include logged messages** check box.
  - -or-
- Select the **Only include user-defined messages** check box.
- 5. Click **Find** to find all messages that match the search criteria.

## How to edit a SQL Server message (Enterprise Manager)

#### To edit a SQL Server message

- 1. Expand a server group, and then right-click a server.
- 2. Point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. Click the **Messages** tab, select the message to edit, and then click **Edit**.

## How to delete a SQL Server message (Enterprise Manager)

#### To delete a SQL Server message

- 1. Expand a server group, and then right-click a server.
- 2. Point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. Click the **Messages** tab, select the message to delete, and then click **Delete**.
- 4. Confirm the deletion.

## How to add a new SQL Server message (Enterprise Manager)

#### To add a new SQL Server message

- 1. Expand a server group, and then right-click a server.
- 2. Point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. On the **Messages** tab, click **New**.
- 4. To specify message options, do the following:
  - In the **New SQL Server Message** dialog box, select an error number for the message.
  - In the **Severity** box, type or select a severity level.
  - In the **Message text** box, type the text for the message.
  - In the **Language** box, click the language to be used in the message. The default is English.
- If you want the message always to be written to the Microsoft® Windows® application log, select the Always write to Windows NT event log check box.

## How to find a SQL Server message (Enterprise Manager)

#### To find a SQL Server message

- 1. Expand a server group, and then right-click a server.
- 2. Point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. Click the **Search** tab, and then specify the text, error number, and severity level for the message. You can also choose to include only logged or user-defined messages.
- 4. Click **Find**.

## How to set up a linked server (Enterprise Manager)

#### To set up a linked server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, right-click **Linked Servers**, and then click **New Linked Server**.
- 3. Click the **General** tab, and in the **Linked server** box, type the name of the server to link.
- 4. Under **Server type**, click a selection.

If you select **Other data source**, you will have to specify provider properties.

## How to delete a linked server (Enterprise Manager) To delete a linked server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then expand **Linked Servers**.
- 3. Right-click the linked server to delete, and then click **Delete**.
- 4. Confirm the deletion.

## How to configure log shipping (Enterprise Manager)

#### To configure log shipping with the Database Maintenance Plan Wizard

**Note** Before you configure log shipping, you must create a share on the primary database to make the transaction logs available. This is a share off of the directory that the transaction logs are dumped to. For example, if you dump the logs to the directory e:\data\tlogs\, you could create the \\logshipping\tlogs share off the directory.

1. In the **Select Databases** screen, select the **These databases** check box, and then select the database to log ship.

If you select more than one database, log shipping will not work, and the log shipping option will not be available. You are not allowed to select a database that is already configured for log shipping.

- 2. Select the **Ship the transaction logs to other SQL Servers (Log Shipping)** check box.
- 3. Continue through the wizard, specifying the rest of the database maintenance options, until you get to the **Specify the Log Shipping Destinations** screen.
- 4. Click **Add** to add a destination database.

For this option to be available, you must have selected to use log shipping earlier in the wizard.

5. In the **Add Destination Database** screen, select a server name.

The server must be registered and running Microsoft® SQL Server<sup>™</sup> 2000 Enterprise Edition to appear in the drop-down list. If you want this destination to become an available source destination, you must select the **Allow database to assume primary role** check box. If this box is not selected, this destination database will not be able to assume the source destination role in the future. If you have selected the **Allow** 

**database to assume primary role** check box, you must also specify the **Transaction Log Backup Directory** on the destination database to which the logs will be backed up.

- 6. To change the transaction log destination database from the default location, enter a location in the **Directory** box.
- 7. If the source database does not exist on the destination database, select the **Create New Database** check box.

The **Database Name** box will default to the source database name. If you want a different database name on the destination server, specify a new name. If you have chosen to allow this destination database to assume the source role, you cannot change the database name from the default.

- 8. If you have selected the **Create New Database** check box, you must specify the file directories for the data and log on the destination database in the **For Data** and **For Log** boxes.
- 9. If the source database already exists on the destination database, select the **Use Existing Database** check box. If the database name on the destination server is different, enter it in the **Database Name** box. This database must have been restored using the WITH STANDBY option to properly accept logs.
- 10. In the **Initialize the Destination Databases** screen, either:
  - Click **Take full database backup now**.

–or-

- Click **Use most recent backup file** to initialize the destination database.
- 11. In the **Log Shipping Schedules** screen, view the default log shipping schedule. If you would like to alter the schedule, click **Change**.

- 12. In the **Copy/Load Frequency** box, set the frequency, in minutes, with which you want the destination servers to backup and restore the transaction logs from the source server.
- 13. In the **Load Delay** box, set the delay, in minutes, you want the destination database to wait before it restores the transaction log from the source server.

The default for this box is 0 minutes, which indicates that the destination database should immediately restore any transaction log backups.

14. In the **File Retention Period** box, specify the length of time that must elapse before a transaction log can be deleted.

## 15. In **the Log Shipping Thresholds** screen, set the **Backup Alert Threshold**.

This is the maximum elapsed time since the last transaction log backup was made on the source server. After the time exceeds this specified threshold, an alert will be generated by the monitor server.

16. In the **Out of Sync Alert** box, specify how long a time has passed between the last transaction log backup on the source server and the last transaction log restore on the destination server.

After the time exceeds this specified threshold, an alert will be generated by the monitor server.

- 17. In the **Specify the Log Shipping Monitor Information** screen, type the name of the server that will monitor log shipping.
- 18. Click either Use Windows Authentication or Use SQL Server Authentication to connect to the monitor server. The log\_shipping\_monitor\_probe login name is fixed and must be used to connect to the monitor server. If this is a new account, choose a new password. If the account already exists on the monitor server, you must specify the existing password.

**Note** Using the Database Maintenance Wizard to set up log shipping, you can log ship only to disks; the backup to tape option is not available.

## How to remove log shipping (Enterprise Manager)

#### To remove log shipping

- 1. Expand a server group, and then expand the primary server.
- 2. Expand Management, and then click **Database Maintenance Plans**.
- 3. In the details pane, right-click the database maintenance plan to delete, and then click **Properties**.
- 4. Click the **Log Shipping** tab, and then click **Remove Log Shipping**.

This stops log shipping on the primary server, removes all secondary servers, and removes the monitor server. You must delete the database maintenance plan to remove additional jobs.

## How to add or edit a destination server (Enterprise Manager) To add or edit a destination server

- 1. Expand a server group, and then expand the primary server.
- 2. Expand Management, and then click **Database Maintenance Plans**.
- 3. In the details pane, right-click the database maintenance plan to edit, and then click **Properties**.
- 4. Click the **Log Shipping** tab, and then click **Add** or **Edit**.
- 5. In the **Directory** box, type or select the directory in which to store the transaction logs.
- 6. Do one of the following:
  - Click **Create New Database** to create a new database on the destination server. If you create a new database, you must specify the database name, along with the file directories for the data and logs.
  - Click **Use Existing Database** if the database already exists on the destination server.

## How to delete a destination server (Enterprise Manager)

#### To delete a destination server

- 1. Expand a server group, and then expand the primary server.
- 2. Expand Management, and then click **Database Maintenance Plans**.
- 3. In the details pane, right-click the database maintenance plan to edit, and then click **Properties**.
- 4. Click the **Log Shipping** tab, select the destination server to delete, and then click **Delete**.
- 5. Confirm the deletion.

**Note** If you delete the only destination server, all of log shipping is removed from the destination, source, and monitor servers. You must use the Database Maintenance Plan Wizard or the **Database Properties** dialog box to add other destination servers.

## How to view the status of servers configured for log shipping (Enterprise Manager)

#### To view the status of servers configured for log shipping

- 1. Expand a server group, and then expand the monitor server.
- 2. Expand Management, and then click **Log Shipping Monitor**.
- 3. In the details pane, right-click the log shipping pair to monitor, and then click **Properties**.
- 4. Click the **Status** tab, and then view information about the status of both the source and destination servers.

Information about the last backup file, the last file copied, and the last file restored is also displayed.

## How to view or edit information about the source server (Enterprise Manager)

#### To view or edit information about the source server

- 1. Expand a server group, and then expand the monitor server.
- 2. Expand Management, and then click **Log Shipping Monitor**.
- 3. In the details pane, right-click the log shipping pair to monitor, and then click **Properties**.
- 4. Click the **Source** tab, and then view or edit information about the source server (for example, the backup failure alert and the alert generation suppression value

There is also an option to view the backup schedule for the source server.

## How to view or edit information about the destination server (Enterprise Manager)

#### To view or edit information about the destination server

- 1. Expand a server group, and then expand the monitor server.
- 2. Expand Management, and then click **Log Shipping Monitor**.
- 3. In the details pane, right-click the log shipping pair to monitor, and then click **Properties**.
- 4. Click the **Destination** tab, and then view or edit information about the destination server (for example, the backup failure alert and the alert generation suppression).

There is also an option to view and/or enable the copy and restore schedules for the destination server.

## How to add an external tool to the Tools menu (Enterprise Manager)

#### To add an external tool to the Tools menu

- 1. On the **Tools** menu, click **External Tools**.
- 2. Click Add.
- 3. Enter a command and parameters.

You can launch any Microsoft® Windows NT® 4.0 or Windows® 2000 application from SQL Server Enterprise Manager. External applications can be added to and run from the **Tools** menu.

**Note** You can use [SVR] and [DBN] in the **Parameters** box to represent the current server and database. When you run the tool, SQL Server Enterprise Manager substitutes the current server and database names, passing them as parameters to the application.

## How to launch SQL Server Enterprise Manager in the Computer Management console

**Note** In Microsoft® Windows® 2000, you can access SQL Server Enterprise Manager from the Computer Management console.

## To launch SQL Server Enterprise Manager in the Computer Management console

- 1. On the **Start** menu, point to **Programs**, point to **Administrative Tools**, and then click **Computer Management**.
- 2. Expand the **Services and Applications** group.
- 3. Expand the **Microsoft SQL Servers** group.

#### See Also

How to enable child windows (Enterprise Manager)

# How to enable child windows in SQL Server Enterprise Manager

**Note** The first time you use the Computer Management console to access SQL Server Enterprise Manager, you may need to change the console mode to enable child windows.

#### To enable child windows in SQL Server Enterprise Manager

- 1. On the **Start** menu, point to **Search**, and then click **For Files or Folders**.
- 2. In the **Search for files or folders named** box, type **Compmgmt.msc**, and then click **Search Now**.
- 3. Right-click **Compmgmt.msc**, and then click **Author**.
- 4. On the **Console** menu, click **Options**.
- 5. In the **Console mode** list, click **User mode limited access, multiple window**.

## See Also

<u>How to launch SQL Server Enterprise Manager in the Computer Management</u> <u>console (Windows)</u>

# How to configure the affinity mask (Enterprise Manager) To configure the affinity mask

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Processor** tab.
- 4. Under **Processor control**, select one or more processors to assemble your affinity mask.

# How to set the allow updates option (Enterprise Manager)

#### To set the allow updates option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Server Settings** tab.
- 4. Under **Server behavior**, select or clear the **Allow modifications to be made directly to the system catalogs** check box.

# How to configure the cost threshold for parallelism (Enterprise Manager)

### To configure the cost threshold for parallelism

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Processor** tab.
- 4. Under **Parallelism**, in the **Minimum query plan threshold for considering queries for parallel execution (cost estimate)** box, type or select a value from 0 through 32767.

This threshold value is relevant only in symmetrical multiprocessing (SMP) environments.

# How to set the default language (Enterprise Manager)

#### To set the default language

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Server Settings** tab.
- 4. In the **Default language for user** box, choose the language in which Microsoft® SQL Server<sup>™</sup> should display system messages.

The default language is English.

# How to set a fixed fill factor (Enterprise Manager)

#### To set a fixed fill factor

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Database Settings** tab.
- 4. Under **Settings**, select the **Fixed** check box, and then position the fill factor slider.

# How to configure the number of processors available for parallel queries (Enterprise Manager)

### To configure the number of processors available for parallel queries

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Processor** tab.
- 4. Under **Parallelism**, select the number of processors to execute queries in parallel.

By default, all available processors are used.

# How to set minimum query memory (Enterprise Manager)

#### To set minimum query memory

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Memory** tab.
- 4. In the **Minimum query memory** box, type or select a value from 512 through 2147483647 kilobytes (KB).

The default value is 1024 KB.

# How to configure the maximum number of worker threads (Enterprise Manager)

### To configure the maximum number of worker threads

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Processor** tab.
- 4. In the **Maximum worker threads** box, type or select a value from 32 through 32767.

The default value is 255.

# How to set the backup retention duration (Enterprise Manager) To set the backup retention duration

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Database Settings** tab.
- 4. In the **Default backup media retention (days)** box, type or select a value from 0 through 365 to set the number of days the backup medium will be retained after a database or transaction log backup.

The default value is 0 days.

# How to set a fixed amount of memory (Enterprise Manager)

# To set a fixed amount of memory

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Memory** tab.
- 4. Click **Use a fixed memory size (MB)**, and then position the fixed memory slider.

**Note** If you use the default settings, Microsoft<sup>®</sup> SQL Server<sup>™</sup> configures memory dynamically.

# How to set the nested triggers option (Enterprise Manager)

#### To set the nested triggers option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Server Settings** tab.
- 4. Under **Server behavior**, select or clear the **Allow triggers to be fired which fire other triggers (nested triggers)** check box.

### See Also

**Using Nested Triggers** 

# How to set the priority boost option (Enterprise Manager)

#### To set the priority boost option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Processor** tab.
- 4. Under **Processor control**, select the **Boost SQL Server priority on Windows** check box.

# How to set the recovery interval (Enterprise Manager)

#### To set the recovery interval

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Database Settings** tab.
- 4. Under **Recovery**, in the **Recovery interval (min)** box, type or select a value from 0 through 32767 to set the maximum amount of time, in minutes, that Microsoft® SQL Server<sup>™</sup> should spend recovering each database at startup.

The default value is 0 minutes, indicating automatic configuration.

## How to set remote server access (Enterprise Manager)

#### To set remote server access

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Connections** tab.
- 4. Under **Remote server connections**, select or clear the **Allow other SQL Servers to connect remotely to this SQL Server using RPC** check box.

# How to enforce distributed transactions for remote procedures (Enterprise Manager)

### To enforce distributed transactions for remote procedures

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Connections** tab.
- 4. Under **Remote server connections**, select the **Enforce distributed transactions (MTS)** check box.

**Note** Remote server connections must be allowed before this value can be set.

# How to set a time limit for remote queries (Enterprise Manager)

#### To set a time limit for remote queries

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Connections** tab.
- 4. Under Remote server connections, in the Query time-out (sec, 0 = unlimited) box, type or select a value from 0 through 2147483647 to set the maximum number seconds that Microsoft® SQL Server<sup>™</sup> will wait before timing out.

**Note** Remote server connections must be allowed before this value can be set.

# How to set the working set size option (Enterprise Manager)

#### To set the working set size option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Memory** tab.
- 4. Select or clear the **Reserve physical memory for SQL Server** check box.

# How to set the two digit year cutoff option (Enterprise Manager) To set the two digit year cutoff option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Server Settings** tab.
- 4. Under **Two digit year support**, in the **When a two-digit year is entered, interpret it as a year between** box, type or select a value that is the ending year of the time span.

The default time span for Microsoft® SQL Server<sup>™</sup> is 1950-2049, which represents a cutoff year of 2049. This means that SQL Server interprets a two-digit year of 49 as 2049, a two-digit year of 50 as 1950, and a two-digit year of 99 as 1999. To maintain backward compatibility, leave the setting at the default value.

Many client applications, such as those based on automation objects, use 2030 as the cutoff year. To make SQL Server compatible with those client applications, specify a time span of 1931-2030.

## How to set user connections (Enterprise Manager)

#### To set user connections

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Connections** tab.
- 4. Under Connections, in the Maximum concurrent user connections

   (0 = unlimited) box, type or select a value from 0 through 32767 to set the maximum amount of simultaneous user connections allowed to the instance of Microsoft® SQL Server<sup>™</sup>.

# How to configure user options (Enterprise Manager)

#### **To configure user options**

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Connections** tab.
- 4. In the **Default connection options** box, select one or more attributes to configure the default query-processing options for all connected users.

By default, no user options are configured.

# How to configure packet size (Enterprise Manager) To configure packet size

- 1. On the **Tools** menu, click **Options**.
- 2. Click the **Advanced** tab.
- 3. In the **Packet size (bytes)** box, type a value.

# How to set the query governor cost limit option (Enterprise Manager)

### To set the query governor cost limit option

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Server Settings** tab.
- 4. Under **Server behavior**, select or clear the **Use query governor to prevent queries exceeding specified cost** check box.

If you select this check box, in the spin box, enter a nonzero, nonnegative value, which the query governor uses to disallow execution of any query with a running length exceeding that value.

# How to enable encryption after SQL Server has been installed (Network Utility)

**Note** If you want to use encryption with a failover cluster, you must install the server certificate with the fully qualified DNS name of the virtual server on all nodes in the failover cluster. For example, if you have a two-node cluster, with nodes named test1.redmond.corp.microsoft.com and test2.redmond.corp.microsoft.com and a virtual SQL Server "Virtsql", you need to get a certificate for "virtsql.redmond.corp.microsoft.com" and install the certificate on both nodes. You can then check the **Force protocol encryption** check box on the Server Network Utility to configure your failover cluster for encryption.

#### To enable encryption

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**
- 2. If **Multiprotocol** does not appear under **Enabled protocols**, click it under **Disabled protocols**, and then click **Enable**. Otherwise, skip to Step 3.
- 3. Under **Enabled protocols**, click **Multiprotocol**, and then click **Properties**.
- 4. Select the **Enable encryption** check box.

### See Also

Multiprotocol Clients

# How to connect to SQL Server through Microsoft Proxy Server (Setup)

**Note** To listen remotely by way of Remote WinSock (RWS), define the local address table (LAT) for the proxy server so that the listening node address is outside the range of LAT entries.

### To connect to SQL Server through Microsoft Proxy Server

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.
- 2. If **TCP/IP** does not appear under **Enabled protocols**, click it under **Disabled protocols**, and then click **Enable**. Otherwise, skip to Step 3.
- 3. Under **Enabled protocols**, click **TCP/IP**, and then click **Properties**.
- 4. Ensure that the port is correct.

# How to set the polling interval (Service Manager)

#### To set the polling interval

- 1. Start SQL Server Service Manager, if it is not already running.
- 2. In the Microsoft® Windows® taskbar, right-click **SQL Server Service Manager**, and then click **Options**.
- 3. In the **Polling interval (seconds)** box, enter a polling interval.

The polling interval determines how often SQL Server Service Manager checks the state of Microsoft SQL Server<sup>™</sup>, SQL Server Agent, and Microsoft Distributed Transaction Coordinator (MS DTC).

4. Optionally, select the **Verify service control action** check box if you want SQL Server Service Manager to provide a confirmation box before stopping, pausing, starting, or continuing a service, including dependent services such as SQL Server Agent.

### See Also

How to set the polling interval (Enterprise Manager)

# How to configure a mail profile (Windows)

#### To configure a mail profile

- Log on to the Microsoft<sup>®</sup> Windows NT<sup>®</sup> 4.0 or Microsoft Windows<sup>®</sup> 2000 server by specifying the name and password used to start Microsoft SQL Server<sup>™</sup> services.
- 2. In **Control Panel**, double-click the **Mail** icon to create and configure or copy an existing mail profile. Select the appropriate mail service to interact with your mail host. Choose a name for the profile that will help the recipient identify the messages.
- 3. On the instance of SQL Server, start the mail client using the newly created mail profile.
- 4. Send a message addressed to the same profile name to ensure that the mail client, mail profile, and e-mail provider are working properly.

If your e-mail message does not appear, you may need to establish mail synchronization by going to the **Tools** menu and then clicking **Deliver Now**.

# How to set up SQL Mail (Enterprise Manager)

#### To set up SQL Mail

- 1. Expand a server group, and then expand a server.
- 2. Expand **Support Services**, right-click **SQL Mail**, and then click **Properties**.
- 3. In the **Profile name** list, type or select the mail profile that you configured for SQL Mail.
- 4. Click **Test** to check the ability of the system to start and stop mail client services with the profile entered.

You should receive a message that SQL Mail has started successfully and stopped a mail session with this profile.

# How to set up SQLAgentMail (Enterprise Manager)

#### To set up SQLAgentMail

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. If you have configured a mail profile using a domain account different from the one used by Microsoft® SQL Server<sup>™</sup>, click **This account**, and then enter the Microsoft Windows NT® 4.0 or Windows 2000® account name and password used to create the mail profile for SQLAgentMail.
- 4. In the **Mail profile** box, select the mail profile you created for SQLAgentMail.
- 5. Click **Test** to check the ability of the system to start and stop mail client services with the profile entered.

You should receive a message that SQL Mail has started successfully and stopped a mail session with this profile.

# How to enable the Lock Page in Memory option (Windows)

**Note** This functionality is available only if you are running the Microsoft® Windows® 2000 operating system.

#### To enable the Lock Page in Memory option

- 1. On the **Start** menu, click **Run**, and then in the **Open** box, type **gpedit.msc**.
- 2. On the **Group Policy** console, expand **Computer Configuration**, and then expand **Windows Settings**.
- 3. Expand **Security Settings**, and then expand **Local Policies**.
- 4. Select the Users Rights Assignment check box.The policies will be displayed in the details pane.
- 5. In the details pane, double-click **Lock pages in memory**.
- 6. In the **Local Security Policy Setting** dialog box, click **Add**.
- 7. In the **Select Users or Groups** dialog box, add an account with privileges to run **sqlservr.exe**.

# How to start the SQL Server Network Utility (Network Utility)

#### To start the SQL Server Network Utility

• On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.

## How to load an installed server network library (Network Utility)

#### To load an installed server network library

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.
- 2. On the **General** tab, select the name of the instance of Microsoft® SQL Server<sup>™</sup> on which to load an installed server network library.
- 3. Under **Disabled protocols**, click the protocol you want to enable, and then click **Enable**.

Depending on which server network library you are loading, you may need to specify any necessary connection parameters. To change any necessary connection parameters, click the protocol and then click **Properties**.

# How to deactivate a server network library configuration (Network Utility)

#### To deactivate a server network library configuration

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.
- 2. Under **Enabled protocols**, click the appropriate server network library configuration, and then click **Disable**.

# How to edit a server network library configuration (Network Utility)

### To edit a server network library configuration

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.
- 2. Under **Enabled protocols**, click the server network library configuration to edit, and then click **Properties**.
- 3. In the dialog box for the specified protocol, change the server name or parameters to edit.

# How to view the installed SQL Server server network libraries (Network Utility)

To view the installed SQL Server server network libraries

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Server Network Utility**.
- 2. Click the **Network Libraries** tab. The installed Microsoft® SQL Server<sup>™</sup> server network libraries will be displayed.

## **Managing Clients**

A client is a front-end application that uses the services provided by a server. The computer that hosts the application is referred to as the client computer. Client software enables computers to connect to an instance of Microsoft® SQL Server<sup>TM</sup> on a network.

SQL Server clients can include applications of various types, such as:

• OLE DB consumers.

These applications use the Microsoft OLE DB Provider for SQL Server or the Microsoft OLE DB Provider for ODBC to connect to and converse with instances of SQL Server. The OLE DB providers serve as intermediaries between an instance of SQL Server and client applications that consume SQL Server data as OLE DB rowsets.

• ODBC applications.

These include client utilities installed with SQL Server, such as SQL Server Enterprise Manager and SQL Query Analyzer, as well as other applications that use the SQL Server ODBC driver to connect to and converse with an instance of SQL Server.

• DB-Library clients, including the SQL Server **isql** command prompt utility and clients written to DB-Library.

Regardless of the type of application, managing a client consists mainly of configuring its connection with the server components of SQL Server. Depending on the requirements of your site, client management can range from little more than entering the name of the server computer to building a library of custom configuration entries to accommodate a diverse multiserver environment.

## How to start the Client Network Utility (Windows)

### To start the Client Network Utility

• On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Client Network Utility**.

# How to display the network library version numbers (Client Network Utility)

### To display the library version numbers

• Click the **Network Libraries** tab.

The network library, library file name, version, file date, and size are displayed.

# How to set DB-Library conversion preferences (Client Network Utility)

**Note** This procedure applies to Microsoft® Windows® 32-bit operating system clients.

### To set the DB-Library conversion preferences

- 1. Click the **DB-Library Options** tab.
- 2. Select or clear the **Automatic ANSI to OEM conversion** check box.
- 3. Select or clear the **Use international settings** check box.

# How to add a network library configuration (Client Network Utility)

### To add a network library configuration

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, under **Network libraries**, select one of the network libraries.
- 3. Enter the server alias and any required parameter information for the network library selected.

# How to edit a network library configuration (Client Network Utility)

### To edit a network library configuration

- 1. Click the **Alias** tab, and then click the network protocol configuration to edit.
- 2. Click **Edit**.
- 3. In the **Edit Network Library Configuration** dialog box, edit the information to change.

# How to delete a network library configuration (Client Network Utility)

### To delete a network library configuration

- 1. Click the **Alias** tab, and then click the network library configuration to delete.
- 2. Click **Remove**.

## How to alias a client to an alternate pipe (Client Network Utility)

### To alias a client to an alternate pipe

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Named Pipes**.
- 3. In the **Server alias** box, enter the server alias.
- 4. Under **Connection parameters**, in the **Pipe name** box, type the name of the alternate pipe name (for example, \\myserver\pipe\altpipe).

## How to configure a client to use the Multiprotocol Net-Library (Client Network Utility)

**Note** Before creating a Multiprotocol client configuration, make sure your computer has at least one IPC protocol loaded under Multiprotocol on the server (Named Pipes, NWLink IPX/SPX, TCP/IP, or Windows Sockets).

### To configure a client to use the Multiprotocol Net-Library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Multiprotocol**.
- 3. In the **Server alias** box, enter the name of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Multiprotocol Net-Library.
- 4. Leave the **Additional parameters** box empty, unless the server requires specific parameters. Verify with your network administrator before entering parameters.

## How to configure a client to use TCP/IP (Client Network Utility)

#### To configure a client to use TCP/IP

- 1. Click the **General** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **TCP/IP**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Windows Sockets Net-Library.

With TCP/IP, you can also specify the server with its IP address instead of its name.

- 4. Do one of the following:
  - Select the **Dynamically determine port** check box to automatically determine the port.
  - Clear the **Dynamically determine port** check box to set the port manually, and then in the **Port number** box, type the port number.

For more information about other TCP/IP protocols that support Windows Sockets, see the TCP/IP documentation.

### How to configure a client to use the NWLink IPX/SPX network library (Client Network Utility)

### To configure a client to use the NWLink IPX/SPX network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **NWLink IPX/SPX**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the NWLink IPX/SPX Net-Library.
- 4. Under **Connection parameters**, click either **Service name** or **Network address**, and then do one of the following:
  - If you clicked **Service name**, enter the service name.

**Service name** is the Microsoft Windows NT® 4.0 or Windows® 2000 computer name under which an instance of SQL Server is running. This name is stored in the Bindery of the server computer.

• If you clicked **Network address**, enter the address (the MAC address), port (socket number), and network (NetWare network number).

## How to configure a client to use the AppleTalk network library (Client Network Utility)

### To configure a client to use the AppleTalk network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **AppleTalk**.
- 3. In the **Server alias** box, enter the name of the instance of Microsoft® SQL Server<sup>™</sup> listening on the AppleTalk Net-Library.
- 4. Under **Connection** parameters, type the AppleTalk object name and optional zone identifiers.

### How to configure a client to use the Banyan VINES network library (Client Network Utility)

### To configure a client to use the Banyan VINES network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Banyan VINES**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Banyan VINES Net-Library.
- 4. Under **Connection** parameters, type the service and the VINES organization. You can use the default value of MSSQL for group.

## How to configure a client to use the VIA network library (Client Network Utility)

To configure a client to use the Banyan VINES network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the Add Network Library Configuration dialog box, click VIA.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the VIA Net-Library.
- 4. Under **Connection parameters**, type the server name and server port number.

### How to configure a client to use a nonstandard network library (Client Network Utility)

To configure a client to use a nonstandard network library

- 1. Click the **Alias** tab, and then click **Add**.
- 2. In the **Add Network Library Configuration** dialog box, click **Other**.
- 3. In the **Server alias** box, enter the alias of the instance of Microsoft® SQL Server<sup>™</sup> listening on the Net-Library you plan to install.
- 4. Under **Connection parameters**, type the file name (file must be a DLL) of the installed Net-Library. Do not enter the DLL extension.
- 5. If necessary, enter any additional information in the **Parameters** box (such as user name and password). Use comma separators between parameters.

## How to verify that SQL Server is listening on AppleTalk and can accept a client connection (Client Network Utility)

To verify that SQL Server is listening on AppleTalk and can accept a client connection

- 1. Copy the client AppleTalk Net-Library (Dbmsadsn.dll) from the \WINNT\system32 directory of the server to the same directory of a remote computer running Microsoft® Windows NT® or Microsoft Windows® 2000 Services for Macintosh.
- 2. On the remote workstation, start **SQL Server Client Configuration**.
- 3. If **AppleTalk** is listed in the **Disabled protocols** list, click **AppleTalk**, and then click **Enable**.
- 4. In the **Enabled protocols by order** list, click **AppleTalk**, and then click the **up** button until **AppleTalk** is at the top of the list.
- 5. Click **OK**.
- 6. Attempt an ISQL connection with the AppleTalk service object name.

For example, at the command line, type:

isql -Usa -P -Sservicename

If you can connect with ISQL and execute queries, the server is configured properly and is accepting connections.

**Note** The Microsoft Win32® AppleTalk (ADSP) client side Net-Library (Dbmsadsn.dll) is included for testing ADSP connections and troubleshooting AppleTalk connections between Macintosh clients and Microsoft SQL Server<sup>TM</sup>. This Net-Library is intended to be used only when testing a connection from a remote client to an instance of SQL Server. If you attempt to make local connections through the ADSP Net-Library to an instance of SQL Server listening on AppleTalk, you will receive the following network error: Net-Library error 11: getsockopt().

## How to check the ODBC SQL Server driver version (Windows)

**Note** You can follow these steps only if you are running the Microsoft® Windows NT® 4.0 operating system.

### To check the ODBC SQL Server driver version (32-bit ODBC)

- 1. In Control Panel, double-click **ODBC Data Sources**.
- 2. Click the **Drivers** tab.

Information for the Microsoft SQL Server<sup>™</sup> entry is displayed in the **Version** column.

## **Automating Administrative Tasks**

This topic defines automated administration and its components, multiserver administration, and introduces Microsoft® SQL Server<sup>™</sup> tools for defining automated tasks.

## What is Automated Administration?

Automated administration is the programmed response to predictable administrative responsibilities or server events. Administrators, application writers, and analysts operating data warehouses can benefit from task automation. To automate administration:

- Establish which administrative responsibilities or server events occur regularly and can be administered programmatically.
- Define a set of jobs and alerts.
- Run the SQL Server Agent service.

## Why Should I Automate Administration?

The job of an administrator entails various administrative duties that do not change from day to day and can be tedious chores. By automating recurring administrative tasks and responses to server events, you free time to perform other tasks that require creativity and lack predictable or programmable responses.

## How to set the service startup account for SQL Server Agent (Enterprise Manager)

### To set the service startup account for SQL Server Agent

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Under **Service startup account**, do one of the following:
  - Click **System account** if your jobs require resources from the local server only.
  - Click **This account** if your jobs require resources across the network, including application resources; if you want to forward events to other Windows® application logs; or if you want to notify operators through e-mail or pagers.
- 4. If you clicked **This account**, enter the Microsoft® Windows NT® 4.0 or Windows 2000 account name that SQL Server Agent will use, and then enter the password for this account in the **Password** box.

You must restart SQL Server Agent before these configuration changes take effect.

# How to set the mail profile for SQL Server Agent (Enterprise Manager)

### To set the mail profile for SQL Server Agent

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Under **Mail session**, select a profile in the **Mail profile** box. If no profiles are listed, enter the name of the profile to use.

## How to set the SQL Server connection (Enterprise Manager)

#### To set the SQL Server connection

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Connection** tab, and under **SQL Server connection**, do of the following:
  - Click **Use Windows Authentication** if you are running SQL Server Agent on a Microsoft® Windows NT® 4.0 or Windows 2000 server.
  - Use SQL Server Authentication if you are running SQL Server Agent on Windows 98.

If you clicked **Use SQL Server Authentication**, in the **SysAdmin login ID** box, enter a login ID of a Microsoft SQL Server<sup>™</sup> login who is a member of the **sysadmin** role. In the **Password** box, enter the password for the SQL Server login.

## How to set a SQL Server alias (Enterprise Manager)

#### To set a SQL Server alias

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Connection** tab, and in the **Local host server** list, click the alias to which SQL Server Agent should connect.

Select an alias that refers to the local instance of Microsoft® SQL Server<sup>™</sup> or SQL Server Agent will not work correctly.

## How to create a job (Enterprise Manager)

#### To create a job

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, and then click **New Job**.
- 4. In the **Name** box, enter a name for the job.
- 5. Clear the **Enabled** check box if you do not want the job to be run immediately following its creation. For example, if you want to test a job before it is scheduled to run, disable the job.
- 6. Under **Source**, do one of the following:
  - Click **Target local server** if the job should run on this server only. Skip to Step 9 if you select this option.
  - Click **Target multiple servers** if the job should run on other servers. Then click **Change**.

This option is enabled only if the server is a master server.

- 7. In the **Change Job Target Servers** dialog box, on the **Available Servers** tab, click a server, and then click the right arrow to move the server to the **Selected target servers** list.
- 8. Click **OK** to return to the **New Job Properties** dialog box.
- 9. In the **Owner** list, select a user to be the owner of the job.

10. In the **Description** box, enter a description of what the job does. The maximum number of characters is 512.

**IMPORTANT** Each job must have at least one step. A step must be created with the job before the job can be saved. For more information, see <u>Creating Job Steps</u>.

## How to disable a job (Enterprise Manager)

### To disable a job

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. Clear the **Enabled** check box.

### How to create a job category (Enterprise Manager)

#### To create a job category

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, point to **All Tasks**, and then click **Manage Job Categories**.
- 4. In the **Job Categories** dialog box, click **Add**.
- 5. In the **Name** box, enter a name for the job category.

#### See Also

How to assign a job to a job category (Enterprise Manager)

## How to delete a job category (Enterprise Manager)

### To delete a job category

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, point to **All Tasks**, and then click **Manage Job Categories**.
- 4. Click a job category, and then click **Delete**.

## How to assign a job to a job category (Enterprise Manager)

### To assign a job to a job category

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. In the **Category** list, select the job category you want to assign to the job.

# How to change the membership of a job category (Enterprise Manager)

### To change the membership of a job category

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, point to **All Tasks** and then click **Manage Job Categories**.
- 4. In the **Job Categories** dialog box, select a job category, and then click **Properties**.
- 5. Click **Show all jobs**.
- 6. In the **Select jobs for this category** list, select or clear the **Member** check box.

## How to give others ownership of a job (Enterprise Manager)

### To give others ownership of a job

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. In the **Owner** list, select a login.

Assigning a job to another login does not guarantee that the new owner has sufficient permission to run the job successfully.

### How to create a CmdExec job step (Enterprise Manager)

#### To create a CmdExec job step

1. Create a new job or right-click an existing job, and then click **Properties**.

For more information about creating a job, see <u>Creating Jobs</u>.

- 2. In the **Job Properties** dialog box, click the **Steps** tab, and then click **New**.
- 3. In the **Step name** box, enter a job step name.
- 4. In the **Type** list, click **Operating system command (CmdExec)**.
- 5. In the **Process exit code of a successful command** box, enter a value from 0 to 999999.
- 6. In the **Command** box, enter the operating system command or executable program.

# How to reset SQLAgentCmdExec permissions (Enterprise Manager)

### To reset SQLAgentCmdExec permissions

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Job System** tab.
- 4. Under Non-SysAdmin job step proxy account, clear the Only users with SysAdmin privileges can execute CmdExe and ActiveScripting job steps check box, and click Reset Proxy Account.
- 5. Type the user name, password, and domain of the user account to be used by SQL Server Agent when running jobs owned by users who are not system administrators.

# How to define Transact-SQL job step options (Enterprise Manager)

### To define Transact-SQL job step options

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. Click the **Steps** tab, click a job step, and then click **Edit**.
- 3. Confirm that the job type is Transact-SQL Script (TSQL), and then click the **Advanced** tab.
- 4. Enter the name of an output file, and decide whether the file should be overwritten or appended to.
- 5. Select the **Append output to step history** check box if you want the output included in the steps history.

Output will only be shown if there were no errors. Also, output may be truncated.

6. In the **Run as user** list, click the user (available to system administrators only).

### How to create an Active Script job step (Enterprise Manager)

#### To create an Active Script job step

1. Create a new job or right-click an existing job, and then click **Properties**.

For more information on creating a job, see <u>Creating Jobs</u>.

- 2. Click the **Steps** tab, and then click **New**.
- 3. In the **Step name** box, enter a job step name.
- 4. In the **Type** list, click **ActiveX Script**.
- 5. In the **Language** list, click a scripting language, or click **Other** and then enter the name of the Microsoft® ActiveX® scripting language in which the command will be written.
- 6. In the **Command** box, enter the source for the job step.
- 7. Click **Parse** to check your syntax.

The message "Parse succeeded" is displayed when your syntax is correct. If an error is found, correct the syntax before continuing.

**Note** There are some instances in which the "Parse succeeded" message is displayed, but the command does not run successfully. To determine whether the command will run successfully, run the command in an ActiveX script authoring environment such as Microsoft Visual Basic®.

## How to set job step success or failure flow (Enterprise Manager) To set job step success or failure flow

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. Click the **Steps** tab, click a step, and then click **Edit**.
- 3. Click the **Advanced** tab.
- 4. In the **On success action** list, click the action to perform if the job step completes successfully.
- 5. In the **Retry attempts** box, enter the number of times from 0 through 9999 that the job step should be repeated before it is considered to have failed. If you entered a value greater than 0 in the **Retry attempts** box, enter in the **Retry interval (minutes)** box the number of minutes from 1 through 9999 that must pass before the job step is retried.
- 6. In the **On failure action** list, click the action to perform if the job step fails.

## How to set up the job history log (Enterprise Manager)

#### To set up the job history log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Job System** tab, select the **Limit size of job history log** check box, and then increase or decrease the maximum number of rows for the job history log.

## How to view the job history (Enterprise Manager)

#### To view the job history

- 1. In the details pane, right-click a job, and then do one of the following:
  - Click **View Job History** if you are viewing the history of a local job.
  - Click **Job Status** if you are viewing the history of a multiserver job.
- 2. If you clicked **Job Status**, in the **Multiserver Job Execution Status** dialog box, click **Job**, click a job name, and then click **View Remote Job History**.
- 3. To update the job history, click **Refresh**.

### How to make a master server (Enterprise Manager)

#### To make a master server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.

## 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Make this a Master**.

A server must be running on Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 to be made a master server. The Make MSX Wizard guides you through the process of making a master server.

## How to make a target server (Enterprise Manager)

#### To make a target server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.

## 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Make this a Target**.

A server must be running on Microsoft® Windows NT® 4.0 or Windows® 2000 to be made a target server. The Make TSX Wizard guides you through the process of making a target server.

## How to enlist a target server from a master server (Enterprise Manager)

#### To enlist a target server from master server

- 1. Expand a server group, and then expand a server configured as a master server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Add Target Servers**.
- 4. Select one or more registered servers, and then click **Enlist**.

## How to defect a target server from a master server (Enterprise Manager)

#### To defect a target server from a master server

- 1. Expand a server group, and then expand a server configured as a target server.
- 2. Expand Management.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Defect from MSX**.
- 4. Click **Yes** to confirm that you want to defect this target server from a master server.

## How to defect multiple target servers from a master server (Enterprise Manager)

#### To defect multiple target servers from a master server

- 1. Expand a server group, and then expand a server configured as a master server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. Click **Post Instructions,** and then in the **Instruction type** list, click **Defect**.
- 5. Under **Recipients**, do one of the following:
  - Click **All target servers** to defect all target servers of this master server. Use this option if you want to completely uninstall the current multiserver administration configuration.
  - Click **These target servers**, and then click the corresponding **Select** box, to defect some but not all target servers of this master server.

# How to view a master SQL Server Agent error log (Enterprise Manager)

#### To view a master SQL Server Agent error log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. On the **Target Server Status** tab, right-click a server, and then click **View SQLServerAgent Error log**.

### How to check the status of a target server (Enterprise Manager)

#### To check the status of a target server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. On the **Target Server Status** tab, right-click a server, and then click **Check SQLServerAgent State**.

## How to schedule a job (Enterprise Manager)

#### To schedule a job

- 1. In the details pane, right-click the job, click **Properties**, click the **Schedules** tab, and then click **New Schedule**.
- 2. In the **Name** box, enter a name for the new schedule.
- 3. Clear the **Enabled** check box if you do not want the schedule to take effect immediately following its creation.
- 4. Under **Schedule type**, do one of the following:
  - Click **Start automatically when SQL Server Agent starts** to start the job when the SQL Server Agent service is started.
  - Click **Start whenever the CPU(s) become idle** to start the job when the CPU(s) reach an idle condition.
  - Click **One time** if you want a schedule to run once. To set the one time schedule, enter values in the **On date** and **At time** boxes.
  - Click **Recurring** if you want a schedule to run repeatedly. To set the recurring schedule, click **Change**, and then complete the **Edit Recurring Job Schedule** dialog box.

## How to set CPU idle time and duration (Enterprise Manager)

#### To set CPU idle time and duration

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Advanced** tab, and then select the **The computer is idle when** check box.
- 4. Under **Idle CPU condition**, do the following:
  - Specify the percentage that the average CPU usage must remain below (across all CPUs).
  - Specify the duration in seconds before the computer is considered idle.

## How to notify an operator of job status (Enterprise Manager)

#### To notify an operator of job status

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. In the **Job Properties** dialog box, click the **Notifications** tab.
- 3. If you want to notify an operator by e-mail, select the **E-mail operator** check box, and then in the list do one of the following:
  - Click **When the job succeeds** to notify the operator when the job completes successfully.
  - Click **When the job fails** to notify the operator when the job completes unsuccessfully.
  - Click **Whenever the job completes** to notify the operator regardless of completion status.
- 4. If you want to notify an operator by pager, select the **Page operator** check box, and then in the list do one of the following:
  - Click **When the job succeeds** to notify the operator when the job completes successfully.
  - Click **When the job fails** to notify the operator when the job completes unsuccessfully.
  - Click **Whenever the job completes** to notify the operator regardless of completion status.
- 5. If you want to notify an operator by **net send**, select the **Net send operator** check box, click an operator, and then do one of the

following:

- Click **When the job succeeds** to notify the operator when the job completes successfully.
- Click **When the job fails** to notify the operator when the job completes unsuccessfully.
- Click **Whenever the job completes** to notify the operator regardless of completion status.

## How to write the job status to the Windows application log (Enterprise Manager)

#### To write the job status to the Windows application log

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. Click the **Notifications** tab.
- 3. Select the **Write to Windows application event log** check box, and then in the list, do one of the following:
  - Click **When the job succeeds** to log the job status when the job completes successfully.
  - Click **When the job fails** to log the job status when the job completes unsuccessfully.
  - Click **Whenever the job completes** to log the job status regardless of completion status.

### How to automatically delete a job (Enterprise Manager)

#### To automatically delete a job

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. Click the **Notifications** tab.
- 3. Select the **Automatically delete job** check box, and then in the list, do one of the following:
  - Click **When the job succeeds** to delete the job status when it has completed successfully.
  - Click **When the job fails** to delete the job when it has completed unsuccessfully.
  - Click **Whenever the job completes** to delete the job regardless of completion status.

# How to set the polling interval for target servers (Enterprise Manager)

#### To set the polling interval for target servers

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. On the **Target Server Status** tab, click **Post Instructions**.
- 5. In the **Instruction type** list, select **Set polling interval**.
- 6. In the **Polling interval** box, enter the number of seconds from 10 through 28,800 that must pass before the target server polls the master server.
- 7. Under **Recipients**, do one of the following:
  - Click **All target servers** if all target servers share the same polling interval.
  - Click **These target servers** if not all target servers share the same polling interval, and then select each target server that will use this polling interval.

## How to start a job (Enterprise Manager)

#### To start a job

- In the details pane, right-click the job, and then do of the following:
  - Click **Start Job** if you are working on a single server, or working on a target server, or running a local server job on a master server.
  - Click **Start Job**, and then click **Start on all targeted servers** if you are working on a master server and want all targeted servers to run the job simultaneously.
  - Click **Start Job**, and then click **Start on specific target servers** if you are working on a master server and want to specify target servers for the job.

In the **Post Download Instructions** dialog box, select the **These target servers** check box, and then select each target server on which this job should run.

## How to stop a job (Enterprise Manager)

#### To stop a job

• In the details pane, right-click the job, and then click **Stop Job**.

For a multiserver job, a STOP instruction for the job is posted to all target servers of the job.

# How to force a target server to poll the master server (Enterprise Manager)

#### To force a target server to poll the master server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. Click a target server, and then click **Force Poll**.

## How to view a job (Enterprise Manager)

#### To view a job

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, expand **SQL Server Agent**, and then click **Jobs**.
- 3. In the details pane, right-click a job, and then click **Properties**.You can only view jobs that you own.

## How to resize the job history log (Enterprise Manager)

#### To resize the job history log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Job System** tab, and then select **Limit size of job history log** check box.
- 4. In the **Maximum job history log size (rows)** box, enter the maximum number of rows the job history log should allow.
- 5. In the **Maximum job history rows per job** box, enter the maximum number of job history rows to allow for a job.

## How to clear the job history log (Enterprise Manager)

#### To clear the job history log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, expand **SQL Server Agent**, and then click **Jobs**.
- 3. In the details pane, right-click a job, and then do one of the following:
  - Click **View job history** if you want to clear the history log of a local job.
  - Click **Job status** if you want to clear the history log of a multiserver job. Click **Job**, click a job name, and then click **View Remote Job History**.
- 4. Click **Clear All**.

## How to modify a job (Enterprise Manager)

#### To modify a job

- 1. In the details pane, right-click the job, and then click **Properties**.
- 2. In the **Job Properties** dialog box, update the job's properties, steps, schedule, and notifications using the corresponding tabs.

Unless you are a member of the **sysadmin** role, you can only modify jobs that you own.

## See Also

How to create a job (Enterprise Manager)

How to notify an operator of job status (Enterprise Manager)

How to schedule a job (Enterprise Manager)

## How to modify the target servers for a job (Enterprise Manager) To modify the target servers for a job

- 1. In the details pane, right-click a job and then click **Properties**.
- 2. On the **General** tab, click **Change**.
- 3. Click one or more servers, and then click the right arrow to select the servers as targets for the job.
- 4. Under **Selected target servers**, choose one or more servers, and then click the left arrow to remove the servers as targets for the job.

## How to modify a target server's location (Enterprise Manager)

#### To modify a target server's location

- 1. Expand a server group, and then expand a server.
- 2. Expand Management.
- 3. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and then click **Manage Target Servers**.
- 4. Right-click a server, and then click **Properties**.
- 5. In the **Location** box, enter a location for the server.

## How to delete a job (Enterprise Manager)

#### To delete a job

• In the details pane, right-click the job, and then click **Delete**.

Unless you are a member of the **sysadmin** role, you can only delete jobs that you own.

### How to synchronize target server clocks (Enterprise Manager)

#### To synchronize target server clocks

- 1. Expand a server group, and then expand a server.
- 2. Right-click **SQL Server Agent**, point to **Multi Server Administration**, and the click **Manage Target Servers**.
- 3. Click **Post Instructions**.
- 4. In the **Instruction type** list, select **Synchronize clocks**.
- 5. Under **Recipients**, do one of the following:
  - Click **All target servers** to synchronize all target server clocks with the master server clock.
  - Click **These target servers** to synchronize certain server clocks, and then select each target server whose clock you want to synchronize with the master server clock.

## How to script jobs using Transact-SQL (Enterprise Manager) To script jobs using Transact-SQL

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, point to **All Tasks**, and then click **Generate SQL Script**.
- 4. In the **File name** box, type a name for the script.
- 5. Under **File format**, do one of the following:
  - Click **MS-DOS Text (OEM)** to save the script in OEM format.
  - Click **Windows Text (ANSI)** to save the script in ANSI format.
  - Click **International Text (Unicode)** to save the script in Unicode format.
- 6. Optionally, under **SQL generation options**, do one or both of the following:
  - Select the **Replace job if it exists** check box to include in the script commands to delete jobs that have the same names as the jobs generated by the script.
  - Select the **Include target servers** check box to include in the script commands to generate target server assignments. This

option is available only when scripting multiserver jobs.

7. In the **TSQL batch separator** box, enter a Transact-SQL batch separator.

### How to create an operator (Enterprise Manager)

#### To create an operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Operators**, and then click **New Operator**.
- 4. In the **Name** box, type the name of the operator.
- 5. To define notification methods for the operator, do one or more of the following:
  - In the **E-mail name** box, enter the operator's e-mail address if the operator will be notified by e-mail.
  - In the **Pager e-mail name** box, enter the pager address of the operator's pager service if the operator will be notified by pager. Also click the days when the operator is available to receive pager notifications.
  - In the **Net send address** box, enter the operator's net send address if the operator will be notified by net send.

#### See Also

How to assign alerts to an operator (Enterprise Manager)

#### How to assign alerts to an operator (Enterprise Manager)

#### To assign alerts to an operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Operators**.
- 4. In the details pane, right-click an operator, click **Properties**, and then click the **Notifications** tab.
- 5. Under **Notifications sent to this operator by**, select one or more of the following check boxes to define the notification method for each alert as necessary: **E-mail**, **Pager**, or **Net send**.
- 6. Select the **Operator is available to receive notifications** check box to enable notifications (of all types) for the operator.

## How to format pager addresses (Enterprise Manager)

#### To format pager addresses

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, click **Properties**, and then click the **Alert System** tab.
- 3. In the **To line** boxes and **CC line** boxes, enter the pager address prefix or suffix. The operator's actual pager address is inserted when a notification is sent.
- 4. In the **Subject** box, enter the subject line prefix or suffix.
- 5. Select the **Include body of e-mail in notification page** check box to include the full e-mail message with the pager message (as opposed to the subject line only).

## How to designate a fail-safe operator (Enterprise Manager)

#### To designate a fail-safe operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, click **Properties**, and then click the **Alert System** tab.
- 3. Under **Fail-safe operator**, in the **Operator** list, click an operator.
- 4. Click the appropriate notification type(s) to specify how the operator will be notified.

# How to create an alert using an error number (Enterprise Manager)

#### To create an alert using an error number

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **Alerts**, and then click **New Alert**.
- 3. In the **Name** box, enter a name for this alert.
- 4. Select the **Enabled** check box to enable the alert to run.
- 5. Click **Error number**, and then type a valid error number for the alert.

When an error number is found in the **sysmessages** table, the error number message text is displayed. Otherwise, "Not a valid error number" is displayed.

**Note** To search for errors by message text, error number, or severity, click the browse (...) button to open the **Manage SQL Server Messages** dialog box.

- 6. In the **Database name** list, click the database to restrict the alert to a specific database.
- 7. In the **Error message contains this text** box, enter a keyword or character string to restrict the alert to a particular character sequence. The maximum number of characters is 100.

#### See Also

How to define the response to an alert

## How to create an alert using severity level (Enterprise Manager) To create an alert using severity level

- 1. Expand a server group, and then expand a server.
- 2. Expand Management, right-click Alerts, and then click New Alerts.
- 3. In the **Name** box, type a name for this alert.
- 4. Select the **Enabled** check box to enable the alert to run.
- 5. Click **Severity**, and then click a severity level.

Severity levels from 19 through 25 send a Microsoft® SQL Server<sup>™</sup> message to the Microsoft Windows® application log and trigger an alert. Events with severity levels less than 19 will trigger alerts only if you have used **sp\_altermessage**, RAISERROR WITH LOG, or **xp\_logevent** to force them to be written to the Windows application log.

- 6. In the **Database name** list, click the database to restrict the alert to a specific database.
- 7. In the **Error message contains this text** box, type a keyword or character string to restrict the alert to a particular character sequence.

The maximum number of characters is 100.

#### See Also

How to define the response to an alert

# How to define the response to an alert (Enterprise Manager) To define the response to an alert

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Alerts**, and in the details pane, right-click an alert.
- 4. Click **Properties**, and then click the **Response** tab.
- 5. Select the **Execute job** check box, and then click a job to execute when the alert occurs.

You can create a new job by clicking **(New Job)**, or modify an existing job by clicking the browse (...) button.

6. Under **Operators to notify**, select one or more of the following check boxes for one or more operators: **E-mail**, **Pager**, or **Net send**.

Be sure that each operator's notification method is valid.

- 7. Select the appropriate **Include alert error text in** check box for the notification method used if you want the alert error text to be sent with the notification.
- 8. Under **Additional notification message to send to operator**, enter additional information for the operator.

The maximum number of characters is 512.

#### See Also

How to assign alerts to an operator (Enterprise Manager)

How to create an operator (Enterprise Manager) How to modify a job (Enterprise Manager)

# How to create a user-defined event error message (Enterprise Manager)

#### To create a user-defined event error message

- 1. Expand a server group.
- 2. Right-click a server, point to **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. Click the **Messages** tab, and then click **New**.
- 4. In the **Severity** list, click the severity level.
- 5. In the **Message text** box, enter the new event message.

The maximum number of characters is 255.

6. Select the **Always write to Windows eventlog** check box to write the event message to the Microsoft® Windows® application log.

# How to edit a user-defined event error message (Enterprise Manager)

#### To edit a user-defined event error message

- 1. Expand a server group.
- 2. Right-click a server, point to **All Tasks**, and then click **Manage SQL Server Messages.**
- 3. On the **General** tab, click **Only include user-defined messages**.

To refine your search, you can specify the search to look for a specified error number, message text, or a severity level. Search results are displayed on the **Messages** tab.

4. On the **Messages** tab, click the message to edit, and then click **Edit**.

In the **Edit SQL Server Message** dialog box, you can edit severity level and message text.

# How to delete a user-defined event error message (Enterprise Manager)

#### To delete a user-defined event error message

- 1. Expand a server group.
- 2. Right-click the server, click **All Tasks**, and then click **Manage SQL Server Messages**.
- 3. On the **General** tab, click **Only include user-defined messages**.

To refine your search, you can specify the search to look for a specified error number, message text, or a severity level. Search results are displayed on the **Messages** tab.

4. On the **Messages** tab, click the message to delete, and then click **Delete**.

## How to disable or reactivate an alert (Enterprise Manager)

#### To disable or reactivate an alert

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Alerts**, and in the details pane, right-click an alert, and then click **Properties**.
- 4. Select or clear the **Enabled** check box.

### See Also

How to delete an alert

# How to designate an events forwarding server (Enterprise Manager)

#### To designate an events forwarding server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, click **Properties**, and then click the **Advanced** tab.
- 3. Under **SQL Server event forwarding**, select the **Forward events to a different server** check box.

Event forwarding is only available on Microsoft® Windows NT® 4.0 and Microsoft Windows® 2000.

- 4. In the **Server** list, click a server, and then do one of the following:
  - Click **Unhandled events** to forward only the events that have not been handled by local alerts.
  - Click **All events** to forward all events regardless of whether they have been handled by local alerts.
- 5. In the **If event has severity of or above** list, click the severity level at which events are forwarded to the selected server.

### How to view information about an operator (Enterprise Manager)

#### To view information about an operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Operators**, and in the details pane, right-click an operator, and then click **Properties**.

The **General** tab displays the notification methods and the pager schedule defined for the operator. The **Notifications** tab displays the notifications the operator receives and the most recent notification attempts.

## How to edit an operator (Enterprise Manager)

#### To edit an operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Operators**, and in the details pane, right-click an operator, and then click **Properties**.

### See Also

How to assign alerts to an operator (Enterprise Manager)

How to create an operator (Enterprise Manager)

## How to change an operator's availability (Enterprise Manager)

#### To change an operator's availability

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Operators**, and in the details pane, right-click an operator, click **Properties**, and then click the **Notifications** tab
- 4. Select or clear the **Operator is available to receive notifications** check box.

### See Also

How to delete an operator

### How to delete an operator (Enterprise Manager)

#### To delete an operator

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Operators**, and in the details pane, right-click an operator, and then click **Delete**.
- 4. If the **Delete Operator** dialog box appears, do one of the following:
  - In the **Reassign to** list, click an operator, and then click **Reassign** if you want another operator to receive the alerts and jobs sent to the deleted operator.
  - Click **Delete without reassigning** if you want to delete the operator without reassigning the alerts and jobs.

#### See Also

How to change an operator's availability

### How to view information about an alert (Enterprise Manager)

#### To view information about an alert

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Alerts**, and in the details pane, right-click an alert, and then click **Properties**.

On the **General** tab, under **History**, you can view the date the alert last occurred, the date the alert was last responded to, and the number of times the alert was triggered since the last time the count was reset. On the **Response** tab, you can view the actions that occur when the alert is triggered.

## How to edit an alert (Enterprise Manager)

#### To edit an alert

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Alerts**, and in the details pane, right-click an alert, and then click **Properties**.
- 4. Update the alert properties on the **General** and **Response** tabs.

## See Also

How to create an alert using an error number (Enterprise Manager) How to create an alert using severity level (Enterprise Manager) How to define the response to an alert (Enterprise Manager)

## How to delete an alert (Enterprise Manager)

#### To delete an alert

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Click **Alerts**, and in the details pane, right-click an alert, and then click **Delete**.
- 4. Confirm the deletion.

### See Also

How to disable or reactivate an alert (Enterprise Manager)

# How to script operators using Transact-SQL (Enterprise Manager)

#### To script operators using Transact-SQL

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Operators**, point to **All Tasks**, and then click **Generate SQL Script**.
- 4. In the **File name** box, type a name for the script.
- 5. Under **File format**, do one of the following:
  - Click **MS-DOS Text (OEM)** to save the script in OEM format.
  - Click **Windows Text (ANSI)** to save the script in ANSI format.
  - Click **International Text (Unicode)** to save the script in Unicode format.
- 6. Under **SQL generation options**, do one or both of the following:
  - Select the **Replace operator if it exists** check box to include in the script commands to that will delete any operators with the same name as jobs generated by the script.
  - Select the **Include notifications sent by alerts to the operator** check box to include in the script commands to

generate alert assignments for the operators generated by the script.

7. Enter a Transact-SQL batch separator.

# How to script alerts using Transact-SQL (Enterprise Manager) To script alerts using Transact-SQL

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Alerts**, point to **All Tasks**, and then click **Generate SQL Script**.
- 4. In the **File name** box, type a name for the script.
- 5. Under **File format**, do one of the following:
  - Click **MS-DOS Text (OEM)** to save the script in OEM format.
  - Click **Windows Text (ANSI)** to save the script in ANSI format.
  - Click **International Text (Unicode)** to save the script in Unicode format.
- 6. Under **SQL generation options**, do one or more of the following:
  - Select the **Replace alert if it exists** check box to include in the script commands to delete any alerts with the same name as jobs generated by the script.
  - Select the **Include notifications sent by alerts to the operator** check box to include in the script commands to generate alert assignments for the operators generated by the

script.

- Select the **Include the name of the job executed by the alert** check box to include in the script commands to provide the name of the job executed by the alert.
- 7. In the **TSQL batch separator** box, enter a Transact-SQL batch separator.

Include the name of the job executed by the alert to have the alerts generated by the script reference their response jobs.

### How to set job execution shutdown (Enterprise Manager)

#### To set job execution shutdown

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, click **Properties**, and then click the **Job System** tab.
- 3. Under **Job execution**, in the **Shutdown time-out interval (seconds)** box, increase or decrease the shutdown time-out interval.

This determines how long SQL Server Agent will wait for executing jobs to finish before SQL Server Agent itself finishes.

### How to autostart SQL Server Agent (Enterprise Manager)

#### To autostart SQL Server Agent

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Click the **Advanced** tab, and then select the **Auto restart SQL Server Agent if it stops unexpectedly** check box.

# How to send SQL Server Agent error messages (Enterprise Manager)

#### To send SQL Server Agent error messages

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Under **Error log**, in the **Net send recipient** box, type the user name or computer name.

This feature is available only in Microsoft® Windows NT® 4.0 and Microsoft Windows® 2000.

# How to view SQL Server Agent error log (Enterprise Manager)

#### To view SQL Server Agent error log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Display Error Log**.
- 3. In the **Type** list, click a type of logged item to filter the log contents.
- 4. Optionally, in the **Containing text** box, enter message text to filter the log contents.
- 5. Click **Apply Filter** if you have selected filter parameters.
- 6. Under **Filtered contents**, view the log contents.

# How to rename a SQL Server Agent error log (Enterprise Manager)

#### To rename a SQL Server Agent error log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Under **Error log**, in the **File name** box, enter the new path and file name, or find it using the browse (...) button.

You can only rename the error log when SQL Server Agent is stopped.

# How to write execution trace messages to the SQL Server Agent error log (Enterprise Manager)

To write execution trace messages to the SQL Server Agent error log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, right-click **SQL Server Agent**, and then click **Properties**.
- 3. Under **Error log**, select the **Include execution trace messages** check box.

Because this option can cause the error log to become large, set it only when investigating a specific SQL Server Agent problem.

# How to schedule a DTS package using the SQLServerAgent service (Enterprise Manager)

To schedule a DTS package using the SQLServerAgent service

- 1. Expand a server, and then expand a server group.
- 2. Expand **Management**, and then expand **SQL Server Agent**.
- 3. Right-click **Jobs**, and then click **New Job**.
- 4. Complete the information on the **General** tab, and then click the **Steps** tab.
- 5. Click **New**, and then complete the information in the **New Job Step** dialog box:
  - For **Type**, click **Operating System Command (CmdExec)** from the list.
  - For **Command**, enter the **dtsrun** command for the package.
  - Click the **Advanced** tab for further job customization options.

#### See Also

How to create a job (Enterprise Manager)

# How to create a SQL Server 7.0 compatible script (Enterprise Manager)

#### To create a SQL Server 7.0 compatible script

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**.
- 3. Right-click a database, point to **All Tasks**, and then click **Generate SQL Script**.
- 4. On the **General** tab, click **Show All** to show all objects to script.
- 5. Select the type of object to script, or select the **Script all objects** check box.
- 6. Click the **Formatting** tab, and then select the **Only script 7.0 compatible features** check box.

This option is only available on Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000.

# **Managing Security**

To ensure that data and objects stored in Microsoft® SQL Server<sup>™</sup> are accessed only by authorized users, security must be set up correctly. Security elements that may have to be set up include authentication modes, logins, users, roles, granting, revoking, and denying permissions on Transact-SQL statements and objects, and data encryption.

# How to set up Windows Authentication Mode security (Enterprise Manager)

#### To set up Windows Authentication Mode security

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. On the **Security** tab, under **Authentication**, click **Windows only**.
- 4. Under **Audit level**, select the level at which user accesses to Microsoft® SQL Server<sup>™</sup> are recorded in the SQL Server error log:
  - **None** causes no auditing to be performed.
  - **Success** causes only successful login attempts to be audited.
  - **Failure** causes only failed login attempts to be audited.
  - All causes successful and failed login attempts to be audited.

#### See Also

**Authentication Modes** 

## How to set up Mixed Mode security (Enterprise Manager)

#### To set up Mixed Mode security

- 1. Expand a server group.
- 2. Right-click a server, and then click **Properties**.
- 3. Click the **Security** tab.
- 4. Under Authentication, click SQL Server and Windows.
- 5. Under **Audit level**, select the level at which user accesses to Microsoft® SQL Server<sup>™</sup> are recorded in the SQL Server error log:
  - **None** causes no auditing to be performed.
  - **Success** causes only successful login attempts to be audited.
  - **Failure** causes only failed login attempts to be audited.
  - All causes successful and failed login attempts to be audited.

#### See Also

**Authentication Modes** 

# How to grant a Windows user or group login access to SQL Server (Enterprise Manager)

To grant a Windows NT 4.0 or Window 2000 user or group login access to SQL Server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, right-click **Logins**, and then click **New Login**.
- 3. In the **Name** box, enter the Microsoft<sup>®</sup> Windows NT<sup>®</sup> 4.0 or Windows<sup>®</sup> 2000 account (in the form DOMAIN\User) to be granted access to Microsoft SQL Server<sup>™</sup>.
- 4. Under Authentication, click Windows Authentication.

#### 5. Optionally:

- In **Database**, click the default database to which the user is connected after logging into an instance of SQL Server.
- In **Language**, click the default language in which messages are displayed to the user.

#### See Also

Adding a Windows NT User or Group

# How to grant a Windows user or group access to a database (Enterprise Manager)

To grant a Windows NT 4.0 or Windows 2000 user or group access to a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user or group will be granted access.
- 3. Right-click **Users**, and then click **New Database User**.
- 4. In the Login name box, type or select the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group name to which database access will be granted.
- 5. Optionally, in **User name**, enter the user name that the login is known by in the database. By default, it is set to the login name.
- 6. Optionally, select database role memberships to be granted to the user or group in addition to **public**, the default.

#### See Also

Granting a Windows NT User or Group Access to a Database

# How to add a SQL Server login (Enterprise Manager) To add a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, right-click **Logins**, and then click **New Login**.
- 3. In **Name**, enter a name for the Microsoft<sup>®</sup> SQL Server<sup>™</sup> login.
- 4. Under Authentication, select SQL Server Authentication.
- 5. Optionally, in **Password**, enter a password.
- 6. Optionally:
  - In **Database**, click the default database to which the login is connected after logging into an instance of SQL Server.
  - In **Language**, click the default language in which messages are displayed to the user.

#### See Also

Adding a SQL Server Login

## How to add a linked server login (Enterprise Manager)

#### To add a linked server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Linked Servers**.
- 3. In the details pane, right-click the linked server to which the login will be added, and then click **Properties**.
- 4. On the **Security** tab, click the local login to add.
- 5. Optionally, select the **Impersonate** check box if the local login should connect to the linked server using its own user security credentials.
- 6. Enter the remote user and remote password with which the local login should connect to the linked server when not using the user's security credentials (**Impersonate** not selected).

### See Also

Establishing Security for Linked Servers

How to set up a linked server (Enterprise Manager)

# How to grant a SQL Server login access to a database (Enterprise Manager)

#### To grant a SQL Server login access to a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the login will be granted access.
- 3. Right-click **Users**, and then click **New Database User**.
- 4. In the **Login name** box, click the Microsoft® SQL Server<sup>™</sup> login to which database access will be granted.
- 5. Optionally, in **User name**, enter the user name that the login is known by in the database. By default, it is set to the login name.
- 6. Optionally, select database role memberships in addition to **public**, the default.

#### See Also

Granting a SQL Server Login Access to a Database

guest User

### How to create a SQL Server database role (Enterprise Manager)

#### To create a SQL Server database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create a role.
- 3. Right-click **Roles**, and then click **New Database Role**.
- 4. In the **Name** box, enter the name of the new role.
- 5. Optionally, click **Add** to add members to the **Standard role** list, and then click a user or users to add.

Only users in the selected database can be added to the role.

#### See Also

Creating User-Defined SQL Server Database Roles

# How to add a member to a SQL Server database role (Enterprise Manager)

#### To add a member to a SQL Server database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to which the user will be added, and then click **Properties**.
- 5. Click **Add**, and then click a user or users to add.

Only users in the selected database can be added to the role.

### See Also

Adding a Member to a Predefined Role

Adding a Member to a SQL Server Database Role

## How to add a member to a fixed server role (Enterprise Manager) To add a member to a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the role, and then click **Properties**.
- 4. On the **General** tab, click **Add**, and then click the logins to add.

### See Also

Adding a Member to a Predefined Role

## How to grant SQL Server login access to a user by using the Create Login Wizard (Enterprise Manager)

To grant SQL Server login access to a user by using the Create SQL Server Login Wizard

- 1. On the **Tools** menu, click **Wizards**.
- 2. In the **Select Wizard** dialog box, expand **Database**, and then doubleclick **Create Login Wizard**.
- 3. Complete the steps in the wizard.

#### See Also

Using the Create Login Wizard

## How to view a SQL Server login or Windows user or group (Enterprise Manager)

To view a SQL Server login or Windows NT 4.0 or Windows 2000 user or group

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to view, and then click **Properties**.

#### See Also

**Viewing Logins** 

### How to view a database user (Enterprise Manager)

#### To view a database user

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user belongs.
- 3. Click Users.
- 4. In the details pane, right-click the user to view, and then click **Properties**.

#### See Also

Viewing Database Users

# How to change the password of a SQL Server login (Enterprise Manager)

### To change the password of a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Password** box, on the **General** tab, enter a new password.
- 5. Confirm the password.

### See Also

**Modifying Logins** 

# How to change the default database of a login (Enterprise Manager)

#### To change the default database of a login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Database** list, on the **General** tab, click the new default database to which the login is connected after logging into an instance of Microsoft® SQL Server<sup>TM</sup>.

### See Also

**Modifying Logins** 

# How to change the default language of a login (Enterprise Manager)

#### To change the default language of a login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to modify, and then click **Properties**.
- 4. In the **Language** list, on the **General** tab, click the new default language in which messages are to be displayed to the user.

#### See Also

Modifying Logins

## How to remove a user or group from a database (Enterprise Manager)

#### To remove a user or group from a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user or group belongs.
- 3. Click **Users**.
- 4. In the details pane, right-click the user or group to remove, and then click **Delete**.
- 5. Confirm the deletion.

#### See Also

### How to remove a SQL Server login (Enterprise Manager)

#### To remove a SQL Server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the login to remove, and then click **Delete**.
- 4. Confirm the deletion.

### See Also

## How to revoke a Windows user or group login access from SQL Server (Enterprise Manager)

To revoke a Windows NT 4.0 or Windows 2000 user or group login access from SQL Server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group to revoke, and then click **Delete**.
- 4. Confirm the deletion.

#### See Also

## How to deny login access to a Windows user or group (Enterprise Manager)

To deny login access to a Windows NT 4.0 or Windows 2000 user or group

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Logins**.
- 3. In the details pane, right-click the Microsoft® Windows NT® 4.0 or Windows® 2000 user or group to deny, and then click **Properties**.
- 4. Under **Authentication**, click **Deny access**.

#### See Also

Denying Login Access to Windows NT Accounts

How to grant a Windows NT user or group login access to SQL Server (Enterprise Manager)

### How to remove a linked server login (Enterprise Manager)

#### To remove a linked server login

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Linked Servers**.
- 3. In the details pane, right-click the linked server to which the linked server login to be removed is mapped, and then click **Properties**.
- 4. On the **Security** tab, under **Local login**, click the linked server login to remove, and then select the blank login at the top of the list.

### See Also

# How to view the roles defined in the current database (Enterprise Manager)

#### To view the roles defined in the current database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to view.
- 3. Click **Roles**.

## See Also

**Viewing Roles** 

### How to view the fixed server roles (Enterprise Manager)

### To view the fixed server roles

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.

### See Also

Viewing Roles

## How to view the members of a database role (Enterprise Manager)

#### To view the members of a database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user belongs.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to view, and then click **Properties** to view members.

#### See Also

# How to remove a user account from a database role (Enterprise Manager)

#### To remove a user account from a database role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the role to which the user account belongs, and then click **Properties**.
- 5. Select the user to remove, and then click **Remove**.

#### See Also

# How to view the members of a fixed server role (Enterprise Manager)

#### To view the members of a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the server role to view, and then click **Properties**.

#### See Also

# How to remove a login from a fixed server role (Enterprise Manager)

#### To remove a login from a fixed server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Security**, and then click **Server Roles**.
- 3. In the details pane, right-click the server role to modify, and then click **Properties**.
- 4. On the **General** tab, select the login to remove, and then click **Remove**.

#### See Also

### How to remove a SQL Server role (Enterprise Manager)

#### To remove a SQL Server role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the role exists.
- 3. Click Roles.
- 4. In the details pane, right-click the role, and then click **Delete**.

**Note** You must drop all role members before you can delete the role. Fixed roles cannot be deleted.

• Confirm the deletion.

#### See Also

Removing a SQL Server Database Role

# How to allow access by granting permissions (Enterprise Manager)

#### To allow access by granting permissions (on an object)

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object on which to grant permissions, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to grant each user.

A check indicates a granted permission. Only permissions applicable to the object are listed.

### See Also

**Granting Permissions** 

## How to grant statement permissions to users within a database (Enterprise Manager)

To grant statement permissions to users within a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users to whom statement permissions will be granted, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to grant each user.

A check indicates a granted permission.

### See Also

**Granting Permissions** 

## How to grant permissions on multiple objects to a user, group, or role (Enterprise Manager)

#### To grant permissions on multiple objects to a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role to which permissions will be granted, click either **Users** or **Roles**.
- 4. In the details pane, right-click the user, group, or role to which permissions will be granted, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all objects**, and then select the permission to grant each object.

A check indicates a granted permission. Only permissions applicable to the object are listed.

#### See Also

**Granting Permissions** 

# How to prevent access by denying permissions (Enterprise Manager)

#### To prevent access by denying permissions (on an object)

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object to which access will be denied, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object to which access will be denied, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to deny each user.

An 'X' indicates a denied permission. Only permissions applicable to the object are listed.

### See Also

**Denying Permissions** 

## How to deny statement permissions from users within a database (Enterprise Manager)

#### To deny statement permissions from users within a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users to whom statement permissions will be denied, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to deny each user.

An 'X' indicates a denied permission.

### See Also

**Denying Permissions** 

## How to deny permissions on multiple objects to a user, group, or role (Enterprise Manager)

#### To deny permissions on multiple objects to a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role to which permissions will be denied, click either **Users** or **Roles**.
- In the details pane, right-click the user or group to which permissions will be denied, point to All Tasks, and then click Manage Permissions. If you are denying permission to a role, right-click the role to which permissions will be denied, click Properties, and then click Permissions.
- 5. Click **List all objects**, and then select the permission to deny for each object.

An 'X' indicates a denied permission. Only permissions applicable to the object are listed.

#### See Also

**Denying Permissions** 

### How to revoke permissions on an object (Enterprise Manager)

#### To revoke permissions on an object

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the object belongs.
- 3. Depending on the type of object to which access will be revoked, click one of the following:
  - Tables
  - Views
  - Stored Procedures
- 4. In the details pane, right-click the object to which access will be revoked, point to **All Tasks**, and then click **Manage Permissions**.
- 5. Click **List all users/user-defined database roles/public**, and then select the permission to revoke from each user.

An empty box indicates a revoked permission. Only permissions applicable to the object are listed.

#### See Also

**Revoking Permissions** 

## How to revoke statement permissions from users in a database (Enterprise Manager)

#### To revoke statement permissions from users in a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database containing the users from whom statement permissions will be revoked, and then click **Properties**.
- 3. On the **Permissions** tab, select the statement permission to revoke from each user.

An empty box indicates a revoked permission.

#### See Also

**Revoking Permissions** 

## How to revoke permissions on multiple objects from a user, group, or role (Enterprise Manager)

#### To revoke permissions on multiple objects from a user, group, or role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the user, group, or role belongs.
- 3. Depending on the type of user, group, or role from which permissions will be revoked, click either **Users** or **Roles**.
- 4. In the details pane, right-click the user or group from which permissions will be revoked, point to All Tasks, and then click Manage Permissions. If you are revoking permission from a role, right-click the role to which permissions will be denied, click Properties, and then click Permissions.
- 5. Click **List all objects**, and then select the permission to revoke for each object.

An empty box indicates a revoked permission. Only permissions applicable to the object are listed.

#### See Also

**Revoking Permissions** 

### How to create an application role (Enterprise Manager)

#### To create an application role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create a role.
- 3. Right-click **Roles**, and then click **New Database Role**.
- 4. In the **Name** box, enter the name of the new application role.
- 5. Under **Database role type,** click **Application role**, and then enter a password.

#### See Also

Establishing Application Security and Application Roles

### How to remove an application role (Enterprise Manager)

#### To remove an application role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which the application role exists.
- 3. Click **Roles**.
- 4. In the details pane, right-click the application role to remove, and then click **Delete**.
- 5. Confirm the deletion.

### See Also

Establishing Application Security and Application Roles

## How to reveal or cancel announcement of SQL Server on a network (Windows)

To reveal or cancel announcement of SQL Server on a network

- 1. In **Control Panel**, double-click **Network**.
- 2. Click the **Services** tab.
- 3. In the **Network Services** list, click **Server**, and then click **Properties**.
- 4. Select **Make Browser Broadcasts to LAN Manager 2.x Clients** to reveal the server, or clear the check box to hide the server.

### See Also

**Revealing SQL Server on a Network** 

## How to grant, deny, or revoke permissions on multiple objects to a user-defined role (Enterprise Manager)

To grant, deny, or revoke permissions on multiple objects to a user-defined role

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database to which the role belongs.
- 3. Click **Roles**.
- 4. In the details pane, right-click the user-defined role to which permissions will be granted, denied, or revoked, and then click **Properties**.
- 5. Under **Names**, click **Permissions**.
- 6. Click **List all objects**, and then select the permission to grant, deny, or revoke on each object.

A checkmark indicates a granted permission; an 'X' indicates a denied permission; and an empty box indicates a revoked permission. Only permissions applicable to the object are listed.

## **Monitoring Server Performance and Activity**

There are a variety of tools and techniques that can be used to monitor Microsoft® SQL Server<sup>TM</sup>. Understanding how to monitor SQL Server can help you:

- Determine whether performance improvements can be made.
- Determine user activity to find out what queries users are issuing and who is connecting to SQL Server.
- Troubleshoot problems.
- Test applications.

## How to start SQL Profiler (Enterprise Manager)

#### **To start SQL Profiler**

- 1. On the **Start** menu, point to **Programs/Microsoft SQL Server**, and then click **Enterprise Manager**.
- 2. On the **Tools** menu, click **SQL Profiler**.

### See Also

Starting SQL Server

# How to view current server activity (Enterprise Manager)

#### To view current server activity

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **Current Activity**.

#### 3. Click **Process Info**.

The current server activity is displayed in the details pane.

## See Also

Monitoring with SQL Server Enterprise Manager Monitoring with Transact-SQL Statements

# How to view the last command batch for a connection (Enterprise Manager)

### To view the last command batch for a connection

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **Current Activity**.
- 3. Click **Process Info**.

The current server activity is displayed in the details pane.

- 4. In the details pane, right-click a Process ID, and then click **Properties**.
- 5. Optionally, click **Refresh** to update the display.

## See Also

Monitoring with SQL Server Enterprise Manager Monitoring with Transact-SQL Statements

## How to view the current locks (Enterprise Manager)

#### To view the current locks

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **Current Activity**.
- 3. Do one of the following:
  - Expand Locks / Process ID to view the current locks for each connection.
  - Expand Locks / Object to view the current locks for each object.
- 4. In the console tree, click the connection (SPID) or object to view.

The current locks for the connection or object are displayed in the details pane.

## See Also

**Displaying Locking Information** 

Monitoring with SQL Server Enterprise Manager

Monitoring with Transact-SQL Statements

**Understanding Locking in SQL Server** 

# How to send a message to a currently connected user (Enterprise Manager)

#### To send a message to a currently connected user

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **Current Activity**.
- 3. Click **Process Info**.

The current server activity is displayed in the details pane.

4. In the details pane, right-click a Process ID, and then click **Send Message**.

**Note** It is not possible to send a message to a user when SQL Server Enterprise Manager is running on Microsoft® Windows® 98.

- 5. In the **Message** box, type the message.
- 6. Optionally, select **Using hostname**, and enter the computer name to send the message to a specific computer.

## See Also

Monitoring with SQL Server Enterprise Manager

## How to terminate a process (Enterprise Manager)

#### To terminate a process

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **Current Activity**.

#### 3. Click **Process Info**.

The current server activity is displayed in the details pane.

- 4. In the details pane, right-click a Process ID, and then click **Kill Process**.
- 5. Confirm that the process has terminated.

# See Also

Monitoring with SQL Server Enterprise Manager

# How to view the SQL Server error log (Enterprise Manager)

#### To view the SQL Server error log

- 1. Expand a server group, and then expand a server.
- 2. Expand **Management**, and then expand **SQL Server Logs**.
- 3. Click the SQL Server Log to view.

Error log details appear in the details pane.

## See Also

Viewing the SQL Server Error Log

## How to start Performance Monitor (Windows)

#### **To start Performance Monitor in Windows NT 4.0**

• On the **Start** menu, point to **Programs/Administrative Tools**, and then click **Performance Monitor**.

## See Also

**Running System Monitor** 

# How to start System Monitor (Windows)

### **To start System Monitor in Windows 2000**

• On the **Start** menu, point to **Programs/Administrative Tools**, and then click **Performance**.

## See Also

**Running System Monitor** 

## How to set up a SQL Server database alert (Windows NT)

#### To set up a SQL Server database alert

- 1. On the **View** menu, click **Alert**.
- 2. On the **Edit** menu, click **Add to Alert**.
- 3. In the **Object** list, click a SQL Server object, and then in the **Counter** box, click a counter on which the alert will be based.
- 4. Under **Alert If**, click either **Over** or **Under**, and then enter a threshold value.

The alert will be generated when the value for the counter is more than or less than the threshold value (depending on whether you click **Over** or **Under**).

5. Click **First Time** or **Every Time** to determine how often the alert is generated.

The default is **Every Time**.

## See Also

Creating a SQL Server Database Alert

## How to set up a SQL Server database alert (Windows)

#### To set up a SQL Server database alert

- 1. On the **Tree** tab of the Performance window, expand **Performance Logs and Alerts**.
- 2. Right-click **Alerts**, and then click **New Alert Settings**.
- 3. In the **New Alert Settings** dialog box, type a name for the new alert, and then click **OK**.
- 4. Click **Add** to add a counter to the alert.

All alerts must have at least one counter.

- 5. In the **Performance Object** list, select a Microsoft® SQL Server<sup>™</sup> object, and then in the **Select counters from list** box, select a counter.
- 6. To add the counter to the alert, click **Add**. You can continue to add counters, or you can click **Close** to return to the **Alert** dialog box.
- 7. In the **Alert** dialog box, click either **Over** or **Under** from the **Alert when value is** list, and then enter a threshold value.

The alert will be generated when the value for the counter is more than or less than the threshold value (depending on whether you clicked **Over** or **Under**).

- 8. In the **Sample data every** boxes, set the sampling frequency.
- 9. On the **Action** tab, set actions to occur every time the alert is triggered.
- 10. On the **Schedule** tab, set the start and stop schedule for the alert scan.

# See Also

Creating a SQL Server Database Alert

## How to view the Windows application log (Windows)

#### To view the Windows application log

- 1. On the **Start** menu, point to **Programs/Administrative Tools**, and then click **Event Viewer**.
- 2. If the Microsoft® Windows® application log is not displayed, on the **Log** menu, click **Application**.

Microsoft SQL Server<sup>™</sup> events are identified by the entry **MSSQLSERVER** in the **Source** column. SQL Server Agent events are identified by the entry **SQLSERVERAGENT**. Microsoft Search service events are identified by the entry **Microsoft Search**.

- 3. To view the log of a different computer, on the **Log** menu, click **Select Computer** and complete the **Select Computer** dialog box.
- 4. Optionally, to display only SQL Server events, on the **View** menu click **Filter Events**, and in the **Source** list, select **MSSQLSERVER**. To view only SQL Server Agent events, select **SQLSERVERAGENT** instead.
- 5. To view more information about an event, double-click the event.

## See Also

How to view the SQL Server error log

Viewing the Windows NT Application Log

## How to enable SQL Server support of SNMP on Windows 98 (SQL Server Network Utility)

#### **To enable SQL Server support of SNMP on Windows 98**

- 1. Install the Microsoft® Windows® 98 SNMP Agent by clicking on the **Network** icon in **Control Panel**. Click **Add**, click **Service** as the type of network component to install, click **Add**, and then click **Have Disk**.
- 2. Install SNMP Agent from Tools\Reskit\Netadmin\Snmp directory.

# How to copy the SQL Server MSSQL-MIB to an SNMP workstation (Windows)

#### To copy the MSSQL-MIB to an SNMP workstation

- 1. Under C:\Program Files\Microsoft SQL Server\MSSQL\Binn, locate the Mssql.mib file.
- 2. Copy the Mssql.mib file to the appropriate directory on the monitoring workstation.
- 3. Repeat these steps for all workstations that will be monitoring Microsoft<sup>®</sup> SQL Server<sup>™</sup>.

For more information about loading the Microsoft SQL Server Management Information Base (MSSQL-MIB) and monitoring SQL Server, see the Simple Network Management Protocol (SNMP) application documentation.

## How to set trace definition defaults (SQL Profiler)

#### To set trace definition defaults

- 1. On the **Tools** menu, click **Options**.
- 2. In the **Trace Options** dialog box, select a template from either the **Template name** list or from the **Template file name** list.
- 3. Select or clear the **Start tracing immediately after making a connection** check box.

## See Also

## How to set trace display defaults (SQL Profiler)

#### To set trace display defaults

- 1. On the **Tools** menu, click **Options**.
- 2. In the **Trace Options** dialog box, click the **Display** tab.
- 3. In the **Font name** list, select the font to be used by SQL Profiler to display traces.
- 4. In the **Font size** list, select a number between 6 and 24.

## See Also

## How to create a trace (SQL Profiler)

#### To create a trace

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **SQL Server** list, select the server to be traced, and then select a connection.
- 3. In the **Trace name** box, type a name for the trace, and then do the following:
  - In the **Trace SQL** Server list, select a server for the trace to run on.
  - In the **Template name** list, select a trace template on which the trace will be based.
- 4. Do one of the following:
  - click **Save to file** to capture the trace to a file in another location.
  - click **Save to table** to capture the trace to a database table.
- 5. Optionally, select the **Enable trace stop time** check box to specify a stop date and time.
- 6. To complete other trace properties, click the **Events**, **Data Columns**, or **Filters** tabs and set the options on these tabs.
- 7. Click **Run** when ready to start the trace.

### See Also

# How to add or remove events from a trace template or trace file (SQL Profiler)

#### To add or remove events from a trace template or trace file

- 1. On the **File** menu, point to **Open**, and then click **Trace Template** or **Trace File**.
- 2. Select the template file or trace file to open.
- 3. In the **Trace Template Properties** dialog box, click the **Events** tab.
- 4. In the **Available events classes** list, expand an event group and click an individual event, or click the entire event group.
- 5. Click **Add** to add the selected event or event group to the events that will be traced.
- 6. In the **Selected event classes** list, expand an event group and click an individual event, or click the entire event group.
- 7. Click **Remove** to remove the selected event or event group from the events that will be traced.

## See Also

# How to add or remove data columns from a trace template (SQL Profiler)

#### To add or remove data columns from a trace template

- 1. On the **File** menu, point to **Open**, and then click **Trace Template**.
- 2. Select the template file to open.
- 3. In the **Trace Template Properties** dialog box, click the **Data Columns** tab.
- 4. In **Unselected data** list, click a data column, and then click **Add** to add the data column to the data that will be captured.
- 5. In the **Selected data** list, click a data column, and then click **Remove** to remove the data column from the data that will be captured.
- 6. In the **Selected data** list, click an individual data column, and then click **Up** or **Down** to order the data column.

## See Also

## How to filter events in a trace template (SQL Profiler)

#### To filter events in a trace template

- 1. On the **File** menu, point to **Open**, and then click **Trace File** or **Trace Template**.
- 2. Select the template file or trace file to open.
- 3. In the **Trace Template Properties** dialog box, click the **Filters** tab.
- 4. In the **Trace event criteria** list, click a criterion.
- 5. Enter a value in the field that appears beneath the trace event criterion.

## How to save trace results to a file (SQL Profiler)

#### To save trace results to a file

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace name** box, type a name for the trace, and then select the **Save to file** check box.
- 4. Set the maximum file size in the **Set maximum file size (MB)** check box. You must set the maximum file size if you are saving trace results to a file.
- 5. Optionally, after saving the file, do the following:
  - Select the **Enable file rollover** check box, which creates new files to store the trace data if the maximum file size is reached. This option is selected by default when you are saving trace results to a file.
  - Select the **Server processes SQL Server trace data** check box.

To avoid missing events, select this option.

## See Also

## How to save trace results to a table (SQL Profiler)

#### To save trace results to a table

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace name** box, type a name for the trace, and then click **Save to table**.
- 4. In the **Destination Table** dialog box, do the following:
  - In the **Database** list, select the destination database.
  - In the **Table** list, type or select the table name for the trace results.
- 5. Select the **Set maximum rows (in thousands)** check box to specify the maximum number of rows to save.

## See Also

# How to modify a trace template (SQL Profiler)

#### To modify a trace template

- 1. On the **File** Menu, point to **Open**, and then click **Trace Template**.
- 2. Select the trace template file to open.
- 3. In the **Trace Template Properties** dialog box, specify the trace template configurations by doing the following:
  - Click the **Events** tab to modify the list of selected event classes.
  - Click the **Data Columns** tab to modify the list of selected data.
  - Click the **Filters** tab to modify the criteria for determining which events to capture. To add or edit criteria, expand the trace event and then type the criteria in the field that appears beneath the trace criterion.

These modifications are effective the next time the trace template is used for a trace.

4. Click **Save**, or click **Save As** to save the trace template under another name.

## See Also

# How to pause a trace (SQL Profiler)

## To pause a trace

- 1. Select the window for a trace that is running.
- 2. On the **File** menu, click **Pause Trace**.

# See Also

# How to run a trace after it has been paused or stopped (SQL Profiler)

## To run a trace after it has been paused or stopped

- 1. Select the window containing the stopped or paused trace.
- 2. On the **File** menu, click **Run Trace**.

## See Also

# How to clear a trace window (SQL Profiler)

#### To clear a trace window

- 1. When multiple traces are active, select the trace window to clear.
- 2. On the **Edit** menu, click **Clear Trace Window**.

The contents of the trace window are removed.

See Also

## How to close a trace window (SQL Profiler)

#### To close a trace window (which closes a trace)

- 1. Select the trace window to close. Closing the trace window closes the trace.
- 2. On the **File** menu, click **Close**.

## See Also

# How to stop a trace (SQL Profiler)

## To stop a trace

- 1. Select a running trace.
- 2. On the **File** menu, click **Stop Trace**, or close a trace window.

# See Also

## How to view filter information (SQL Profiler)

#### To view filter information

- 1. Create a trace, or open a trace template, trace file, or SQL Script.
- 2. If you created a trace, on the **File** menu, click **Properties**.
- 3. Click the **Filters** tab to view the filter information.

## See Also

# How to open a trace data file (SQL Profiler)

## To open a trace data file

- 1. On the **File** menu, point to **Open**, and then click **Trace File**.
- 2. Select the trace data file to open.

# See Also

# How to open a trace table (SQL Profiler)

#### To open a trace table

- 1. On the **File** menu, point to **Open**, and then click **Trace Table**.
- 2. In the **Connect to SQL Server** dialog box, select the instance of Microsoft® SQL Server<sup>™</sup> that contains the trace table and a connection method.
- 3. In the **Source Table** dialog box, in the **Database** list, click the database in which the table is saved.
- 4. In the **Table** list, click the table name.

## See Also

# How to replay a trace table (SQL Profiler)

#### To replay a trace table

- 1. On the **File** menu, point to **Open**, and then click **Trace Table**.
- 2. In the **Connect to SQL Server** dialog box, select the instance of Microsoft® SQL Server<sup>™</sup> that contains the trace table and a connection method.
- 3. In the **Source Table** dialog box, in the **Database** list, click the database in which the table is saved.
- 4. On the **Replay** menu, click **Start**.
- 5. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 6. In the **Replay SQL Server** dialog box, select the destination server, and then select any of the following:
  - Replay events in the order they were traced
  - Replay events using multiple threads
  - Display replay results

## See Also

# How to replay a trace file (SQL Profiler)

#### To replay a trace file

- 1. On the **File** menu, point to **Open**, and then click **Trace File**.
- 2. Select the file to open.
- 3. On the **Replay** menu, click **Start**.
- 4. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 5. In the **Replay SQL Server** dialog box, select the destination server, and then select any of the following:
  - Replay events in the order they were traced
  - Replay events using multiple threads
  - Display replay results

## How to replay a single event at a time (SQL Profiler)

#### To replay a single event at a time

- 1. On the **File** menu, point to **Open**, and then click **Trace File** or **Trace Table**.
- 2. If you choose to replay an event from a trace table, you must enter the connection information. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method, and specify the database in which the table is saved in the **Source Table** dialog box.
- 3. Select the trace file to open, unless you have already selected the trace table.
- 4. On the **Replay** menu, click **Step**.
- 5. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 6. In the **Replay SQL Server** dialog box, alter any necessary settings, and then click **Start**.
- 7. On the **Replay** menu, click **Step**.
- 8. Repeat Step 6 until you have replayed all the necessary event steps.

#### See Also

## How to replay to a breakpoint (SQL Profiler)

#### To replay to a breakpoint

- 1. On the **File** menu, point to **Open**, and then click **Trace File** or **Trace Table**.
- 2. If you choose to replay a trace table, you must enter the connection information. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method, and specify the database in which the table is saved in the **Source Table** dialog box.
- 3. Select the trace file to open, unless you have already selected the trace table.
- 4. In the trace windows, click a trace event.
- 5. On the **Replay** menu, click **Toggle Break-Point**.
- 6. On the **Replay** menu, click **Start**.
- 7. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 8. In the **Replay SQL Server** dialog box, alter any necessary settings, and then click **Start**.

### See Also

## How to replay to the cursor (SQL Profiler)

#### To replay to the cursor

- 1. On the **File** menu, point to **Open**, and then click **Trace File** or **Trace Table**.
- 2. If you choose to replay a trace table, you must enter the connection information. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method, and specify the database in which the table is saved in the **Source Table** dialog box.
- 3. Select the trace file to open, unless you have already selected the trace table.
- 4. In the trace window, click an event.
- 5. On the **Replay** menu, click **Run To Cursor**.
- 6. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 7. In the **Replay SQL Server** dialog box, alter any necessary settings and then click **Start**.

#### See Also

## How to replay an SQL script (SQL Profiler)

#### To replay an SQL script

- 1. On the **File** menu, point to **Open**, and then click **SQL Script**.
- 2. Select the Transact-SQL script file to open.
- 3. On the **Replay** menu, click **Start**.
- 4. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 5. In the **Replay SQL Server** dialog box, alter any necessary settings, and then click **Start**.

### See Also

## How to modify a filter (SQL Profiler)

## To modify a filter

- 1. On the **File** menu, point to **Open**, and then click **Trace Template**.
- 2. Select the trace template to open.
- 3. In the **Trace Template Properties** dialog box, click the **Filters** tab.
- 4. In the **Trace event criteria** list, click a criterion.
- 5. Enter a value in the field that appears beneath the criterion.

## See Also

## How to set a maximum file size for a trace file (SQL Profiler)

#### To set a maximum file size for a trace file

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace** name box, type a name for the trace.
  - In the **Template name** list, select a trace template.
- 4. Select **Save to file**, and then specify a file in which to store the trace information.
- 5. In the **Set maximum file size (MB)** check box, specify a maximum file size for the trace. File rollover is enabled by default.

When the file size reaches this maximum, the trace events are no longer recorded.

#### See Also

## How to set a maximum table size for a trace table (SQL Profiler)

#### To set a maximum table size for a trace table

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace Properties** dialog box, in the **Trace** name box, type a name for the trace.
- 4. Select the **Save to table** check box, and then specify a table in which to store the trace information.

You will be prompted to connect to the server on which you want the trace to be stored.

- 5. In the **Destination Table** dialog box, in the **Database** list, select a database for the trace, and then in the **Table** box, type or select a table name.
- 6. Select the **Set maximum rows (in thousands)** check box and specify a maximum number of rows for the trace table.

When the number of rows in the table exceeds this maximum, the trace events are no longer recorded.

### See Also

## How to set an immediate start time for traces (SQL Profiler)

#### To set an immediate start time for traces

- 1. On the **Tools** menu, click **Options**.
- 2. Select the **Start tracing immediately after making a connection** check box.

## See Also

## How to set a StartTime filter for a trace (SQL Profiler)

#### To set a StartTime filter for a trace

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace name** box, type a name for the trace.
  - In the **Template** name list, select a trace template.
  - Optionally, specify a save destination for the trace results.
- 4. Click the **Filters** tab, and then in the **Trace event criteria** box, expand **StartTime**.
- 5. Expand **Greater** or **Less than**, and then enter a time value in the field that appears beneath the criterion.

### See Also

## How to set an EndTime filter for a trace (SQL Profiler)

#### To set an EndTime filter for a trace

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to connect to and a connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace name** box, type a name for the trace.
  - In the **Template name** list, select a trace template.
  - Optionally, specify a save destination for the trace results.
- 4. Click the **Filters** tab, and then in the **Trace event criteria** box, expand **EndTime**.
- 5. Expand **Greater** or **Less than**, and then enter a time value in the field that appears beneath the criterion.

### See Also

## How to filter system IDs in a trace (SQL Profiler)

#### To filter system IDs in a trace

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace name** box, type a name for the trace.
  - In the **Template name** list, select a trace template.
  - Optionally, specify a save destination for the trace results.
- 4. Click the **Filters** tab, and then in the **Trace event criteria** box, expand **SPID**.
- 5. Expand **Equals**, **Not equal to**, **Greater than or equal** or **Less than or equal**, and then enter a value in the field that appears beneath the criterion.

### See Also

## How to create a trace template (SQL Profiler)

#### To create a trace template

- 1. On the **File** menu, point to **New**, and then click **Trace Template**.
- 2. Specify template properties by clicking options on the **Events**, **Data Columns**, or **Filters** tabs.
- 3. Click **Save As** to name and save the template.

## See Also

How to derive a template from a running trace

How to derive a template from a trace file or trace table

## How to set global trace options (SQL Profiler)

#### To set global trace options

- 1. On the **Tools** menu, click **Options**.
- 2. From the **Template name** list, select a default trace template.
- 3. Optionally, select the **Start tracing immediately after making a connection** check box.
- 4. Click the **Display** tab, and then in the **Font Name** list, select the font used by SQL Profiler to display traces.
- 5. In the **Font size** list, type or select a number that ranges from 6 through 24.

### See Also

## How to find a value or data column while tracing (SQL Profiler)

#### To find a value or data column while tracing

- 1. Create a trace by clicking the **File** menu and pointing to **New** and then **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. To display data in the trace window, run SQL Query Analyzer, and then click the trace window in SQL Profiler again.
- 4. On the **Edit** menu, click **Find**.
- 5. In the **Find** dialog box, enter a search value or specify a data column.

#### 6. Click **Find Previous**.

The search starts at the beginning of the trace.

### See Also

# How to launch a new trace with the current template (SQL Profiler)

#### To launch a new trace with the current template

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to be traced and a connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace name** box, type the name of the new trace.
  - In the **Template name** list, select the same template as the currently running trace.

## See Also

## How to derive a template from a running trace (SQL Profiler)

#### To derive a template from a running trace

- 1. On the **File** menu, point to **New**, and then click **Trace**.
- 2. In the **Connect to SQL Server** dialog box, select the server to be traced and connection method.
- 3. In the **Trace Properties** dialog box, do the following:
  - In the **Trace name** box, type the name of the new trace.
  - In the **Template name** list, select a template.
- 4. Modify the template by adding or deleting events, columns, or filters, and then start the trace by clicking **Run**.
- 5. On the **File** menu, point to **Save As**, and then click **Trace Template**.
- 6. Type a name and save the template file.

### See Also

# How to derive a template from a trace file or trace table (SQL Profiler)

#### To derive a template from a trace file or trace table

- 1. On the **File** menu, point to **Open**, and then click either **Trace File** or **Trace Table**.
- 2. If you choose to replay a trace table, you must enter the connection information. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method, and specify the database in which the table is saved in the **Source Table** dialog box.
- 3. Select the trace file to open, unless you have already selected a trace table to open.
- 4. On the **File** menu, point to **Save As**, and then click **Trace Template**.
- 5. Type a name and save the template file.

### See Also

## How to save a template, trace file, or trace table to SQL Script (SQL Profiler)

To save a template, trace file, or trace table to SQL Script

- 1. On the **File** menu, point to **Open**, and then click **Trace Template**, **Trace File**, or **Trace Table**.
- 2. If you choose to save a trace table, you must enter the connection information. In the **Connect to SQL Server** dialog box, select the server to which you want to connect and a connection method.
- 3. In the **Open** dialog box, select a trace template, trace file, or trace table to save.
- 4. On the **File** menu, point to **Save As**, and then click **SQL Script**.

## See Also

## How to create an SQL Script for a running trace (SQL Profiler)

#### To create an SQL Script for running a trace

- 1. Create and run a new trace, or create or open a trace template.
- On the File menu, point to Script Trace, and then click For SQL Server 2000 or For SQL Server 7.0, depending on the server that is to be traced.

#### See Also

## **Creating and Maintaining Databases**

Designing your Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 database structure involves creating and maintaining a number of interrelated components.

Database component	Description
<u>Databases</u>	Contain the objects used to represent, manage, and
	access data.
<u>Tables</u>	Store rows of data and define the relationships
	between multiple tables.
<u>Indexes</u>	Optimize the speed of accessing the data in the
	table.
<u>Views</u>	Provide an alternate way of looking at the data in
	one or more tables.
Stored Procedures	Centralize business rules, tasks, and processes
	within the server using Transact-SQL programs.
<u>Triggers</u>	Centralize business rules, tasks, and processes
	within the server using special types of stored
	procedures that are only executed when data in a
	table is modified.

Creating a database to serve your business needs requires an understanding of how to design, create, and maintain them to ensure your database performs optimally.

## Databases

A database in Microsoft® SQL Server<sup>™</sup> 2000 consists of a collection of tables with data, and other objects, such as views, indexes, stored procedures, and triggers, that are defined to support the activities performed with the data. Before objects within the database can be created, you must create the database and understand how to change the settings and the configuration of the database. This includes tasks such as expanding or shrinking the database, or specifying the files used to create the database.

## See Also

**Databases** 

## How to create a database (Enterprise Manager)

#### To create a database

- 1. Expand a server group, and then expand a server.
- 2. Right-click **Databases**, and then click **New Database**.
- 3. Enter a name for the new database.

The primary database and transaction log files are created using the database name you specified as the prefix, for example newdb\_Data.mdf and newwdb\_Log.ldf. The initial sizes of the database and transaction log files are the same as the default sizes specified for the **model** database. The primary file contains the system tables for the database.

- 4. To change the default values for the new primary database file, click the **General** tab. To change the defaults for the new transaction log file, click the **Transaction Log** tab.
- 5. To change the default values provided in the **File name**, **Location**, **Initial size (MB)**, and **File group** (not applicable for the transaction log) columns, click the appropriate cell to change and enter the new value.
- 6. To specify how the file should grow, select from these options:
  - To allow the currently selected file to grow as more data space is needed, select **Automatically grow file**.
  - To specify that the file should grow by fixed increments, select **In megabytes** and specify a value.

- To specify that the file should grow by a percentage of the current file size, select **By percent** and specify a value.
- 7. To specify the file size limit, select from these options:
  - To allow the file to grow as much as necessary, select **Unrestricted filegrowth**.
  - To specify the maximum size the file should be allowed to grow to, select **Restrict filegrowth (MB)** and specify a value.

**Note** The maximum database size is determined by the amount of disk space available and the licensing limits determined by the version of Microsoft® SQL Server<sup>™</sup> you are using.

## See Also

Creating a Database

**Creating Filegroups** 

## How to create a database using the Create Database Wizard (Enterprise Manager)

#### To create a database using the Create Database Wizard

- 1. Expand a server group, and then expand the server in which to create a database.
- 2. On the **Tools** menu, click **Wizards**.
- 3. Expand **Database**.
- 4. Double-click Create Database Wizard.
- 5. Complete the steps in the wizard.

## See Also

Creating a Database

## How to increase the size of a database (Enterprise Manager)

#### To increase the size of a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to increase, and then click **Properties**.
- 3. To increase the data space, click the **General** tab. To increase the transaction log space, click the **Transaction Log** tab.
- 4. To add a new file, click the next empty row and, in the **File name** column, enter the file name that will contain the additional space.

The file location is generated automatically and given the .ndf suffix for a database file, or an .ldf suffix for a transaction log file.

5. To change the default values provided in the **File name**, **Location**, **Space allocated (MB)**, and **Filegroup** (not applicable for the transaction log) columns, click the cell to change and enter the new value.

For existing files, only the **Space allocated (MB)** value can be changed; the new value must be larger than the existing value.

- 6. To specify how the file should grow, select from these options:
  - To allow the currently selected file to grow as more data space is needed, select **Automatically grow file**.
  - To specify that the file should grow by fixed increments, select **In megabytes** and specify a value.
  - To specify that the file should grow by a percentage of the

current file size, select **By percent** and specify a value.

- 7. To specify the file size limit, select from these options:
  - To allow the file to grow as much as necessary, select **Unrestricted filegrowth**.
  - To specify the maximum size to which the file should be allowed to grow, select **Restrict filegrowth (MB)** and specify a value.

**Note** The maximum database size is determined by the amount of disk space available and the licensing limits determined by the version of SQL Server you are using.

## See Also

Adding and Deleting Data and Transaction Log Files

**Creating Filegroups** 

Expanding a Database

## How to shrink a database (Enterprise Manager)

#### To shrink a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to shrink, point to **All Tasks**, and then click **Shrink Database**.
- 3. To specify how much to shrink the database, select from these options:
  - For **Maximum free space in files after shrinking**, enter the amount of free space you want left in the database after shrinking. Use the **Database Size**, **Space free** value as a guideline.
  - Select **Move pages to beginning of file before shrinking** to cause the freed file space to be retained in the database files, and pages containing data to be moved to the beginning of the database files.
- 4. Click **Schedule** to create or change the frequency or time when the database is automatically shrunk.
- 5. Click **Shrink files** if you want to shrink individual database files.

**Note** You cannot shrink a database smaller than the size of the **model** database.

#### See Also

Shrinking a Database

# How to delete data or log files from a database (Enterprise Manager)

#### To delete data or log files from a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database from which to delete the data or log files, and then click **Properties**.
- 3. To delete data files, click the **General** tab. To delete log files, click the **Transaction Log** tab.
- 4. In the **File name** column, click the arrow next to the name of the file to delete, and then press the DELETE key. A cross will appear next to the file name indicating that the file will be deleted.

**Note** Files can be deleted only if they are empty. Remove all objects on the files and shrink the database before deleting files from the database.

### See Also

Adding and Deleting Data and Log Files

# How to change the configuration settings for a database (Enterprise Manager)

#### To change the configuration settings for a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to change, and then click **Properties**.
- 3. Click the **Options** tab, and select or clear the configuration setting(s) to change.

## See Also

Nested Triggers Renaming a Database Setting Database Options Shrinking a Database

## How to view a database (Enterprise Manager)

#### To view a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then click the database to view.
- 3. Click **General**, **Tables and Indexes**, or **Space Allocated** to view more information about the database.

### See Also

Viewing a Database

## How to view the settings for a database (Enterprise Manager)

#### To view the settings for a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to view, and then click **Properties**.
- 3. Click the **Options** tab.

## See Also

Viewing a Database

## How to view a list of databases on a server (Enterprise Manager)

#### To view a list of databases on a server

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases** and a list of all databases on the server will be displayed.

## See Also

Viewing a Database

## How to display data and log space information for a database (Enterprise Manager)

To display data and log space information for a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then click the database to view.
- 3. In the details pane, click **Space Allocated** to view database space information.

## See Also

Displaying Database and Transaction Log Space Monitoring with Transact-SQL Statements

## How to generate a script (Enterprise Manager)

#### To generate a script

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to script, point to **All Tasks**, and then click **Generate SQL Scripts**.
- 3. On the **General** tab, select the database objects to script. By default, all objects in the database are scripted.
- 4. On the **Formatting** tab, select from the script formatting options:
  - Generate the CREATE <object> command for each object so that each object to be scripted is explicitly created using its existing definition. This is selected by default.
  - Generate the DROP <object> command for each object so that a DROP statement is added to the script for each object to be scripted. This is selected by default.

**CAUTION** When executed, this causes any existing objects in the database (where the script is executed with the same name as objects listed in the script) to be deleted first.

- **Generate scripts for all dependent objects** so that all objects in the database which are needed to create the objects listed in the script are included automatically in the script if not already selected.
- **Include descriptive headers in the script files** so that a comment is added to the file for each object listed in the script.

- 5. On the **Options** tab, select the security-related, table-related, and script file-related options.
- 6. On the **General** tab, click **Preview** to view a preview of the generated script.

## See Also

Documenting and Scripting Databases

# How to start the Database Maintenance Plan Wizard (Enterprise Manager)

### To start the Database Maintenance Plan Wizard

- 1. Expand a server group, and then expand the server.
- 2. On the **Tools** menu, click **Database Maintenance Planner**.
- 3. Complete the steps in the wizard.

## See Also

Database Maintenance Plan Wizard

# How to delete a database (Enterprise Manager)

#### To delete a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, right-click the database to delete, and then click **Delete**.
- 3. Confirm the deletion.

# See Also

**Deleting a Database** 

# How to attach and detach a database (Enterprise Manager) To attach a database

- 1. Expand a server group, and then expand a server.
- 2. Right-click **Databases**, and select **All Tasks/Attach Database**.
- 3. Enter the name of the MDF (**master** data file) of the database to attach. If you are not sure where the file is located, click browse (...) to search. There can only be up to 16 file names specified. For more information, see <u>sp\_attach\_db</u>.
- 4. To ensure that the specified MDF file is correct, click Verify. The Original File Name(s) column lists all the files in the database (data files and log files). The Current File(s) Location column lists the file names and paths. If Microsoft® SQL Server<sup>™</sup> cannot find the files in the specified locations, the attach operation fails. The Current File(s) Location column can be edited, and the current location of the file must be in this column for the attach operation to work. For example, if you have changed the default location of the file before you detached it, you must specify the current location for the attach operation to be successful.
- 5. In the **Attach as** box, enter the name of the database. The database name must not match any existing database names.
- 6. Specify the database owner.
- 7. Click **OK**. A database node for the newly attached database is created in the Database folder.

#### To detach a database

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases.
- 3. Right-click the database, and then select **All Tasks/Detach Database**. This menu is visible only if you are a member of the **sysadmin** fixed server role and the server to which you are connected is SQL Server 2000. The **master**, **model**, and **tempdb** databases cannot be detached.
- 4. In the **Detach Database** dialog box, check the status of the database. To successfully detach a database, STATUS should be: **The database is ready to be detached**. Optionally, you can select to update statistics prior to the detach operation.
- 5. To terminate any existing connections from the database, click **Clear**.
- 6. Click **OK**. The database node for the detached database is removed from the Database folder.

# How to create user-defined data types (Enterprise Manager)

#### To create a user-defined data type

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create the user-defined data type.
- 3. Right-click **User Defined Data Types**, and then click **New User Defined Data Type**.
- 4. Enter the name of the new data type.
- 5. In the **Data type** list, select the base data type.
- 6. If **Length** is active, enter another value if you want to change the maximum data length that the data type can store. The only data types that can have variable lengths are **binary**, **char**, **nchar**, **nvarchar**, **varbinary**, and **varchar**.
- 7. To allow the data type to accept null values, select **Allow Nulls**.
- 8. Optionally, in the **Rule** and **Default** lists, select a rule or default, if any, to bind to the user-defined data type.

# See Also

Creating User-Defined Data Types

sp\_addtype

**Using Data Types** 

# How to delete user-defined data types (Enterprise Manager)

#### To delete a user-defined data type

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database, and then click **User Defined Data Types**.
- 3. In the details pane, right-click the data type to delete, and then click **Delete**.
- 4. To see how deleting this data type will affect the database, click **Show Dependencies**.
- 5. In the **Drop Objects** dialog box, click **Drop All**.

# See Also

Creating User-defined Data Types

<u>sp\_droptype</u>

Using Data Types

# How to create a reflexive relationship (Enterprise Manager)

#### To create a reflexive relationship

- 1. Open a database diagram.
- 2. Click the row selector for the database column that you want to relate to another column.
- 3. While the pointer is positioned over the row selector, drag the pointer outside the table until a line appears.
- 4. Drag the line back to the selected table.
- 5. Release the mouse button.

The **Create Relationship** dialog box appears and attempts to match the primary key columns with the non-key columns to which you dragged the line.

6. Confirm that the columns you want to relate are shown in the **Primary key table** and **Foreign key table** lists.

#### See Also

Drawing a Reflexive Relationship

# How to create a many-to-many relationship between tables (Enterprise Manager)

#### To create a many-to-many relationship between tables

- 1. Open a database diagram.
- 2. Add the tables that you want to create a many-to-many relationship between.
- 3. Create a third table by right-clicking within the database diagram, and then clicking **New Table**.

This will become the junction table.

4. In the **Choose Name** dialog box, enter a name for the table.

For example, the junction table between the titles table and the authors table is named **titleauthors**.

5. Copy the primary key columns from each of the other two tables to the junction table.

You can add other columns to this table, just as you can to any other table.

- 6. In the junction table, set the primary key to include all the primary key columns from the other two tables.
- 7. Define a one-to-many relationship between each of the two primary tables and the junction table.

### See Also

Adding Tables to a Diagram

Copying Columns from One Table to Another

Mapping Many-to-Many Relationships to a Database Diagram

# How to delete a relationship (Enterprise Manager)

#### To delete a relationship

- 1. Open a database diagram.
- 2. Right-click the relationship line that you want to delete from the diagram, and then click **Delete Relationship from Database**.

### See Also

Deleting a Relationship

# How to create a DEFAULT object (Enterprise Manager)

#### To create a DEFAULT object

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create the DEFAULT object.
- 3. Right-click **Defaults**, and then click **New Default**.
- 4. In **Name**, enter a name for the DEFAULT object.
- 5. In **Value**, enter the value for the DEFAULT object. The value can be a constant, expression, or variable.
- 6. Optionally, click:
  - **Bind UDTs** to bind the new DEFAULT object to a userdefined data type.
  - **Bind Columns** to bind the DEFAULT object to an existing column in a table.

#### See Also

Creating and Modifying DEFAULT Definitions

# How to delete a DEFAULT object (Enterprise Manager)

#### To delete a DEFAULT object

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the DEFAULT object belongs, and then click **Defaults**.
- 3. In the details pane, right-click the DEFAULT object to delete, and then click **Delete**.
- 4. To see how deleting this table will affect the database, click **Show Dependencies**.
- 5. Click **Drop All**.

#### See Also

Creating and Modifying DEFAULT Definitions

# How to view the dependencies of a table (Enterprise Manager)

#### To view the dependencies of a table

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table belongs, and then click **Tables**.
- 3. In the details pane, right-click the table, point to **All Tasks**, and then click **Display Dependencies**.

# See Also

Viewing a Table

# Indexes

To create efficient indexes that improve the performance of your database application by increasing the speed of your queries, you need an understanding of how to create and maintain the indexes on the tables in your database.

### See Also

<u>Indexes</u>

# How to analyze a query using Index Analysis (Query Analyzer)

#### To start the Index Tuning Wizard

- 1. Expand a server group, and then expand the server in which to create the index.
- 2. On the **Tools** menu, click **Wizards**.
- 3. Expand **Management**.
- 4. Double-click Index Tuning Wizard.
- 5. Complete the steps in the wizard.

#### See Also

**Index Tuning Wizard** 

# How to analyze a query using Index Tuning Wizard (Query Analyzer)

#### To analyze a query using Index Tuning Wizard

- 1. Enter the query or batch of Transact-SQL statements to be analyzed into the query pane.
- 2. On the **Query** menu, click **Index Tuning Wizard**.

# See Also

Index Tuning Wizard

# How to create an index using the Create Index Wizard (Enterprise Manager)

#### To create an index using the Create Index Wizard

- 1. Expand a server group, and then expand the server in which to create the index.
- 2. On the **Tools** menu, click **Wizards**.
- 3. Expand **Database**.
- 4. Double-click **Create Index Wizard**.
- 5. Complete the steps in the wizard.

# See Also

Creating an Index

# How to view all indexes in a database (Enterprise Manager)

#### To view all indexes in a database

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then click the database to view.
- 3. Click **Tables & Indexes** to view information about the indexes in the database.

#### See Also

Viewing an Index

# Views

By creating, modifying, and maintaining views, you can customize each user's perception of the database.

# See Also

<u>Views</u>

# How to create a view using the Create View Wizard (Enterprise Manager)

#### To create a view using the Create View Wizard

- 1. Expand a server group, and then expand the server in which to create the view.
- 2. On the **Tools** menu, click **Wizards**.
- 3. Expand **Database**.
- 4. Double-click **Create View Wizard**.
- 5. Complete the steps in the wizard.

### See Also

Creating a View

# How to rename a view (Enterprise Manager)

#### To rename a view

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the view belongs, and then click Views.
- 3. In the details pane, right-click the view, and then click **Rename**.
- 4. Enter the new name of the view.
- 5. Confirm the new name.

# See Also

Modifying and Renaming a View

# How to modify a view (Enterprise Manager)

#### To modify a view

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the view belongs, and then click **Views**.
- 3. In the details pane, right-click the view, and then click **Design View**.
- 4. To add additional tables or views to the view, right-click in the diagram pane, and then click **Add Table**.
  - On the **Tables** or **Views** tabs, click the table or view to add to the new view, and then click **Add**. Repeat for each table or view you want to add to the new view.
- 5. To remove an entire table or view from the view, in the diagram pane, right-click the title bar of the table, and then click **Remove**.
- 6. In the **Column** box of the grid pane, select the columns to be referenced in the view.
- 7. Select **Output** if the column is to appear in the result set of the view.
- 8. To group by column, right-click the column, and then click **Group By**.
- 9. In the **Criteria** column, enter the criteria specifying which rows to retrieve; this determines the WHERE clause. If **Group By** is specified, this determines the HAVING clause.

- 10. In the **Or** column, enter any additional criteria to specify which rows to retrieve.
- 11. Right-click anywhere in the grid pane, and then click **Properties**.
- 12. Optionally, select:
  - **Output all columns** to display all columns in the view in the result set.
  - **DISTINCT values** to filter out duplicate values in the result set.
  - **Encrypt view** to encrypt the definition of the view.
- 13. Optionally, in **Top**, enter the number of rows to return in the result set. Enter the word PERCENT after the number to return a percentage of rows in the result set.
- 14. Right-click anywhere in the diagram pane, and then click **Run** (to view the result set) or **Save** (to save the view).

#### See Also

Modifying and Renaming a View

# How to get information about a view (Enterprise Manager)

#### To get information about a view

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the view belongs, and then click Views.
- 3. In the details pane, right-click the view, and then click **Properties**.

# See Also

**Getting Information About a View** 

# How to display the dependencies of a view (Enterprise Manager)

# To display the dependencies of a view

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the view belongs, and then click Views.
- 3. In the details pane, right-click the view, point to **All Tasks**, and then click **Display Dependencies**.

# See Also

**Getting Information About a View** 

# How to delete a view (Enterprise Manager)

#### To delete a view

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the view belongs, and then click **Views**.
- 3. In the details pane, right-click the view, and then click **Delete**.
- 4. To see how deleting this view will affect the database, click **Show Dependencies**.
- 5. Click **Drop All**.

# See Also

**Deleting a View** 

# **Stored Procedures**

By creating, modifying, and using stored procedures, you can simplify your business applications and improve application and database performance.

# See Also

**Stored Procedures** 

# How to create a stored procedure (Enterprise Manager)

#### To create a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the database in which to create the procedure.
- 3. Right-click **Stored Procedures**, and then click **New Stored Procedure**.
- 4. Enter the text of the stored procedure. Press TAB to indent the text of a stored procedure. Press CTRL+TAB to exit the text box, or click an appropriate button.
- 5. To check the syntax, click **Check Syntax**.
- 6. To set the permissions, click **Permissions**.

### See Also

Creating a Stored Procedure

# How to create a stored procedure using the Create Stored Procedure Wizard (Enterprise Manager)

#### To create a stored procedure using the Create Stored Procedure Wizard

- 1. Expand a server group, and then expand the server in which to create the view.
- 2. On the **Tools** menu, click **Wizards**.
- 3. Expand **Database**.
- 4. Double-click Create Stored Procedure Wizard.
- 5. Complete the steps in the wizard.

#### See Also

Creating a Stored Procedure

# How to add an extended stored procedure (Enterprise Manager)

#### To add an extended stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the **master** database.
- 3. Right-click **Extended Stored Procedures**, and then click **New Extended Stored Procedure**.
- 4. In **Name**, enter the name of the extended stored procedure.
- 5. In **Path**, enter the path of the dynamic link library that contains the extended stored procedure. Optionally, click (...) to locate the DLL containing the extended stored procedure.

#### See Also

Creating a Stored Procedure

**Extended Stored Procedures** 

# How to modify a stored procedure (Enterprise Manager)

#### To modify a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the procedure belongs, and then click Stored Procedures.
- 3. In the details pane, right-click the stored procedure, and then click **Properties**.
- 4. In the **Text** box, change the text of the stored procedure as necessary. Press CTRL+TAB to indent the text of a SQL Server Enterprise Manager stored procedure.
- 5. To check the syntax, click **Check Syntax**.
- 6. To change the permissions, click **Permissions**.

### See Also

Modifying and Renaming a Stored Procedure

# How to rename a stored procedure (Enterprise Manager)

#### To rename a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the procedure belongs, and then click **Stored Procedures**.
- 3. In the details pane, right-click the stored procedure, and then click **Rename**.
- 4. Type the new name of the stored procedure.
- 5. Confirm the new name.

#### See Also

Modifying and Renaming a Stored Procedure

# How to view the definition of a stored procedure (Enterprise Manager)

#### To view the definition of a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the stored procedure belongs, and then click **Stored Procedures**.
- 3. In the details pane, right-click the stored procedure, and then click **Properties**.

#### See Also

Viewing a Stored Procedure

# How to view the dependencies of a stored procedure (Enterprise Manager)

#### To view the dependencies of a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the stored procedure belongs, and then click Stored Procedures.
- 3. In the details pane, right-click the stored procedure, point to **All Tasks**, and then click **Display Dependencies**.

#### See Also

Viewing a Stored Procedure

# How to view information about an extended stored procedure (Enterprise Manager)

#### To view information about an extended stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the **master** database, and then click **Extended Stored Procedures**.
- 3. In the details pane, right-click the extended stored procedure, and then click **Properties**.
- 4. Optionally, click (...) to locate the DLL containing the extended stored procedure.
- 5. Optionally, click **Permissions** to view or set permissions on the extended stored procedure.

#### See Also

Viewing a Stored Procedure

## How to delete a stored procedure (Enterprise Manager)

#### To delete a stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the stored procedure belongs, and then click **Stored Procedures**.
- 3. In the details pane, right-click the stored procedure to delete, and then click **Delete**.
- 4. To see how deleting this stored procedure will affect the database, click **Show Dependencies**.
- 5. Click **Drop All**.

#### See Also

**Deleting a Stored Procedure** 

# How to delete an extended stored procedure (Enterprise Manager)

#### To delete an extended stored procedure

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the **master** database, and then click **Extended Stored Procedures**.
- 3. In the details pane, right-click the extended stored procedure to delete, and then click **Delete**.
- 4. To see how deleting this extended stored procedure will affect the database, click **Show Dependencies**.
- 5. Click **Drop All**.

#### See Also

**Deleting a Stored Procedure** 

# Triggers

By understanding how to create, modify, and maintain triggers, you can use triggers to:

- Cascade changes through related tables in the database.
- Disallow or roll back changes that violate referential integrity, thereby canceling the attempted data modification transaction.
- Enforce restrictions that are more complex than those defined with CHECK constraints.
- Find the difference between the state of a table before and after a data modification and take action(s) based on that difference.

### See Also

Enforcing Business Rules with Triggers

### How to create a trigger (Enterprise Manager)

#### To create a trigger

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table to contain the trigger belongs, and then click **Tables**.
- 3. In the details pane, right-click the table on which the trigger will be created, point to **All Tasks**, and then click **Manage Triggers**.
- 4. In **Name**, click **<new>**.
- 5. In the **Text** box, enter the text of the trigger. Use CTRL-TAB to indent the text of a trigger.
- 6. To check the syntax, click **Check Syntax**.

#### See Also

<u>Creating a Trigger</u>

## How to modify a trigger (Enterprise Manager)

#### To modify a trigger

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table containing the trigger belongs, and then click **Tables**.
- 3. In the details pane, right-click the table on which the trigger exists, point to **All Tasks**, and then click **Manage Triggers**.
- 4. In **Name**, select the name of the trigger.
- 5. Change the text of the trigger in the **Text** field as necessary. Press CTRL+TAB to indent the text of a SQL Server Enterprise Manager trigger.
- 6. To check the syntax of the trigger, click **Check Syntax**.

### See Also

Modifying and Renaming a Trigger

## How to view a trigger (Enterprise Manager)

#### To view a trigger

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table containing the trigger belongs, and then click **Tables**.
- 3. In the details pane, right-click the table on which the trigger exists, point to **All Tasks**, and then click **Manage Triggers**.

### See Also

Viewing a Trigger

## How to view the dependencies of a trigger (Enterprise Manager)

#### To view the dependencies of a trigger

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the trigger belongs, and then click **Tables**.
- 3. In the details pane, right-click the table to which the trigger belongs, point to **All Tasks**, and then click **Display Dependencies**.
- 4. In **Object**, click the name of the trigger whose dependencies you want to display.

### See Also

Modifying and Renaming a Trigger

Viewing a Trigger

## How to delete a trigger (Enterprise Manager)

#### To delete a trigger

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table containing the trigger belongs, and then click **Tables**.
- 3. In the details pane, right-click the table on which the trigger exists, point to **All Tasks**, and then click **Manage Triggers**.
- 4. In **Name**, click the name of the trigger to delete.
- 5. Click **Delete**.
- 6. Confirm the deletion.

### See Also

Deleting a Trigger

# **Full-text Indexes**

Full-text support for Microsoft® SQL Server<sup>™</sup> 2000 data requires two tasks: enabling the database to allow queries against character data, and the creation and maintenance of the underlying indexes that facilitate these queries.

# How to enable a database for full-text indexing (Enterprise Manager)

#### To enable a database for full-text indexing

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then click a database to enable.
- 3. On the **Tools** menu, click **Full-Text Indexing**.
- 4. Complete the Full-Text Indexing Wizard.

### See Also

# How to enable a table for full-text indexing (Enterprise Manager)

#### To enable a table for full-text indexing

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, expand the database in which the table belongs, and then click Tables.
- 3. In the details pane, right-click the table, click **Full-Text Index Table**, and then click **Define Full-Text Indexing on a Table**.
- 4. Complete the Full-Text Indexing Wizard.

#### See Also

# How to enable a column for full-text indexing (Enterprise Manager)

#### To enable a column for full-text indexing

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, expand the database in which the table belongs, and then click Tables.
- 3. In the details pane, right-click the table, click **Full-Text Index Table**, and then click **Define Full-Text Indexing on a Table**.
- 4. Complete the Full-Text Indexing Wizard to enable specific columns.

#### See Also

## How to edit a full-text index on a table (Enterprise Manager)

#### To edit a full-text index on a table

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, expand the database in which the table belongs, and then click Tables.
- 3. In the details pane, right-click the table, click **Full-Text Index Table**, and then click **Edit Full-Text Indexing**.
- 4. Make the changes in the Full-Text Indexing Wizard.

#### See Also

## How to remove full-text indexing on a table (Enterprise Manager)

#### To remove a full-text index on a table

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, expand the database in which the table belongs, and then click Tables.
- 3. In the details pane, right-click the table, click **Full-Text Index Table**, and then click **Remove Full-Text Indexing**.
- 4. Click **Yes** to confirm the removal of the full-text index from the table.

#### See Also

### How to create a full-text catalog (Enterprise Manager)

#### To create a full-text catalog

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then right-click the database where you want the full-text catalog.
- 3. Click **New**, and then click **New Full-Text Catalog**.
- 4. Complete the **New Full-Text Catalog** dialog box.

#### See Also

### How to rebuild a full-text catalog (Enterprise Manager)

#### To rebuild a full-text catalog

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the full-text catalog to rebuild.
- 3. Click **Full-Text Catalogs**, and then right-click the specific catalog to rebuild.
- 4. Select **Rebuild Catalog**.
- 5. Click **Yes** to rebuild the catalog.

# See Also

# How to rebuild all full-text catalogs in a database (Enterprise Manager)

#### To rebuild all full-text catalogs

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the catalogs to rebuild.
- 3. Right-click **Full-Text Catalogs**, and then click **Rebuild All Catalogs**.
- 4. Click **Yes** to rebuild all the catalogs.

#### See Also

### How to start and stop a full or incremental population of a fulltext index (Enterprise Manager)

#### To start and stop the production of a full-text index

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the full-text catalog to rebuild.
- 3. Click **Full-Text Catalogs**, and then right-click the specific catalog to populate.
- 4. Click **Start Full Population** or **Start Incremental Population**. Or click **Stop Population**, as appropriate.

#### See Also

# How to check the status, tables, and schedules of a full-text catalog (Enterprise Manager)

#### To check the status, tables, and schedules of a full-text catalog

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the full-text catalog to review.
- 3. Click **Full-Text Catalogs**, and then right-click the specific catalog to review.
- 4. Click **Properties**, and then click the **Status**, **Tables**, and **Schedules** tabs, as appropriate.

#### See Also

<u>Full-Text Indexes</u> <u>sp\_fulltext\_catalog</u>

# How to change or create a new schedule for a full-text catalog (Enterprise Manager)

#### To change or create a new schedule for a full-text catalog

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the full-text catalog to review.
- 3. Click **Full-Text Catalogs**, and then right-click the specific catalog to review.
- 4. Click **Schedules** and make changes or establish a new schedule.

#### See Also

Full-Text Indexes sp\_add\_job sp\_add\_jobschedule sp\_add\_jobserver sp\_fulltext\_catalog sp\_delete\_job sp\_update\_job

# How to remove a full-text catalog from a database (Enterprise Manager)

#### To remove a full-text catalog

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the full-text catalog to rebuild.
- 3. Click **Full-Text Catalogs**, and then right-click the specific catalog you want to remove, and click **Delete**.
- 4. Click **Yes** to remove the catalog.

#### See Also

# How to remove all full-text catalogs in a database (Enterprise Manager)

#### To remove all full-text catalogs in a database

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the catalogs to remove.
- 3. Right-click **Full-Text Catalogs**, and then click **Remove All Catalogs**.
- 4. Click **Yes** to confirm the removal of the catalogs.

#### See Also

# How to repopulate all full-text catalogs for a database (Enterprise Manager)

#### To repopulate all full-text catalogs in a database

- 1. Expand a server group, and then expand a server.
- 2. Expand Databases, and then expand the database that contains the catalogs to repopulate.
- 3. Right-click **Full-Text Catalogs**, and then click **Repopulate All Catalogs**.
- 4. Click **Yes** to repopulate all the catalogs.

#### See Also

# How to clean up the full-text catalogs on a server (Enterprise Manager)

#### To clean up the full-text catalogs on a server

- 1. Expand a server group, and then expand a server.
- 2. Expand Support Services, right-click **Full-Text Search**, and then click **Clean Up Catalogs**.
- 3. Click **Yes** to clean up all full-text catalogs on the server.

#### See Also

# How to start and stop the Microsoft Search Service for full-text support (Enterprise Manager)

If necessary, the full-text service can be started (and stopped) in one of these ways:

- 1. Expand a server group, and then expand a server.
- 2. Expand Support Services, right-click Full-Text Search, and then click **Start (**or **Stop)**.
- 3. You can also start and stop the service by:
  - Selecting the Microsoft Search Service in SQL Server Service Manager and clicking start or stop.
  - Typing **net start mssearch** (or **net stop mssearch**) from a command prompt.

## See Also

# **Accessing and Changing Data**

SQL Server Enterprise Manager includes a tool for designing queries interactively using a graphical user interface. These queries are used:

- In views.
- In Data Transformation Services (DTS) Packages.
- To display the data in Microsoft<sup>®</sup> SQL Server<sup>™</sup> tables.

### How to access the Query Designer in Data Transformation Services (Enterprise Manager)

**To access the Query Designer in Data Transformation Services** 

- 1. Right-click an **Execute SQL Task** object, and then click **Properties**.
- 2. In the Execute SQL Properties window, click **Build Query**.

The Query Designer will open with the diagram and SQL panes visible. To open panes, see <u>Query and View Designer Layout</u>.

**Note** There must be a valid data source connection in the package to build a query. If there is not, a connection must be created before trying to access the Query Designer.

# **Optimizing Database Performance**

The goal of optimizing database performance is to minimize the response time for each query and to maximize the throughput of the entire database server by minimizing network traffic, disk I/O, and CPU time. Understanding how to design the logical and physical structure of the data, tune queries, and configure Microsoft® SQL Server<sup>™</sup> 2000 and the operating system can help optimize database performance.

# **Database Design**

There are two components to designing a database: logical and physical. It is important to understand how to design the database to model your business requirements correctly and to take advantage of hardware and software features early on in the development of a database application. It is difficult to make changes to these components later in the development cycle.

# How to place an existing table on a different filegroup (Enterprise Manager)

#### To place an existing table on a different filegroup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table belongs, and then click **Tables**.
- 3. In the details pane, right-click the table, and then click **Design Table**.
- 4. Right-click any column, and then click **Properties**.
- 5. On the **Tables** tab, in the **Table Filegroup** list, select the filegroup on which to place the table.
- 6. Optionally, in the **Text Filegroup** list, select a filegroup on which to place any **text**, **image**, and **ntext** columns.

#### See Also

**Placing Tables on Filegroups** 

# How to place an existing index on a different filegroup (Enterprise Manager)

### To place an existing index on a different filegroup

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, expand the database in which the table containing the index belongs, and then click **Tables**.
- 3. In the details pane, right-click the table, and then click **Design Table**.
- 4. Right-click any column, and then click **Properties**.
- 5. On the **Indexes/Keys** tab, in the **Selected index** list, select the index to move.
- 6. In the **Index Filegroup** list, select a filegroup on which to place the index.

### See Also

**Placing Indexes on Filegroups** 

# **Query Tuning**

Query tuning involves monitoring and determining if and why a query is not performing as optimally as possible, and then taking steps to resolve any problems. Understanding how to create and update column statistics and indexes can significantly improve query performance.

## How to create statistics (Query Analyzer)

#### **To create statistics**

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that suggests that statistics need to be created (table name in red), and then click **Create Missing Statistics**. The database, table, and column(s) that the Graphical Execution Plan suggests need new statistics are automatically selected.
- 5. Optionally, in **Statistics name**, enter the name for the statistics.
- 6. Optionally, in **Amount of data to sample**, select:
  - **Default** to let Microsoft<sup>®</sup> SQL Server<sup>™</sup> determine the number of rows to sample automatically.
  - **Sample all the data** to instruct SQL Server to sample all of the data in the table.
  - **Sample % of the data** and enter a percentage of data to sample to base the statistics on.
- 7. Optionally, select **Do not automatically recompute statistics (not recommended)** to prevent SQL Server from updating statistics automatically as the data is updated.

8. Optionally, click **Edit SQL** to view and edit the Transact-SQL statement used to create or update the statistics.

# See Also

<u>Graphically Displaying the Execution Plan Using SQL Query Analyzer</u> <u>Statistical Information</u>

# How to update statistics (Query Analyzer)

### To update statistics

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that suggests that statistics need to be updated, and then click **Manage Statistics**. The Graphical Execution Plan automatically selects the appropriate database and table.
- 5. Optionally, in **Database** and **Table**, click the name of a different database and table on which to update the statistics.
- 6. Click **Update**.
- 7. In **Name**, select the statistics to be updated.
- 8. Optionally, in **Amount of data to sample**, select:
  - **Default** to let Microsoft® SQL Server<sup>™</sup> determine the number of rows to sample automatically.
  - **Sample all the data** to instruct SQL Server to sample all of the data in the table.
  - **Sample % of the data** and enter a percentage of data to

sample to base the statistics on.

- **Sample rows** and enter the number of rows to sample to base the statistics on.
- 9. Optionally, in **Update statistics options**, select:
  - **Include columns** to update statistics on columns as well as indexes.
  - **Do not automatically recompute statistics (not recommended)** to prevent SQL Server from updating statistics automatically as the data is updated.
- 10. Optionally, click **Edit SQL** to view and edit the Transact-SQL statement used to create or update the statistics.

## See Also

<u>Graphically Displaying the Execution Plan Using SQL Query Analyzer</u> <u>Statistical Information</u>

## How to delete statistics (Query Analyzer)

#### **To delete statistics**

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that represents the table containing the statistics that need to be deleted, and then click **Manage Statistics**.
- 5. Optionally, in **Database** and **Table**, click the name of a different database and table on which to delete the statistics.
- 6. In **Existing statistics**, click the name of the statistic to delete, and then click **Delete**.
- 7. Confirm the deletion.

### See Also

<u>Graphically Displaying the Execution Plan Using SQL Query Analyzer</u> <u>Statistical Information</u>

### How to create a new index (Query Analyzer)

#### To create a new index

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that suggests that an index needs to be created, and then click **Manage Indexes**. The Graphical Execution Plan automatically selects the appropriate database and table. Click **New**.
- 5. In **Index name**, enter the name for the index.
- 6. In **Column**, select the column to appear in the index. Composite indexes can be created by selecting more than one column.
- 7. Optionally, select a column, and then click either **Move Up** or **Move Down** to change the order of the columns in the index.
- 8. Optionally, in **Index options**, select:
  - **Unique values** to create a unique index.
  - **Clustered index** to create a clustered index. If a clustered index already exists, this option is not available.
  - **Ignore duplicate values** to control what happens when an

INSERT statement inserts multiple, nonunique key values into an index. For more information, see <u>CREATE INDEX</u>.

- **Do not recompute statistics (not recommended)** to specify that index statistics are not automatically recomputed as the index is updated.
- **Filegroup** to specify the filegroup on which to create the index. Click the name of the filegroup.
- **Pad index** to leave space open on each interior node of the index. For more information, see <u>CREATE INDEX</u>.
- **Drop existing** to delete any existing index of the same name before creating the new index.
- **Fill factor** to specify how full SQL Server should make the leaf level of each index page during index creation. For more information, see <u>CREATE INDEX</u>.
- 9. Optionally, click **Edit SQL** to view and edit the Transact-SQL statement used to create the index.

### See Also

Graphically Displaying the Execution Plan Using SQL Query Analyzer

**Placing Indexes on Filegroups** 

# How to modify an index (Query Analyzer)

#### To modify an index

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that suggests that an index needs to be modified, and then click **Manage Indexes**. The database and table that the Graphical Execution Plan suggests need an index modified are automatically selected.
- 5. Optionally, in **Database** and **Table**, select the name of a different database and table.
- 6. In **Existing indexes**, click the name of the index to modify, and then click **Edit**.
- 7. In **Column**, select the column you want to appear in the index. Composite indexes can be created by selecting more than one column.
- 8. Optionally, select a column, and then click either **Move Up** or **Move Down** to change the order of the columns in the index.
- 9. Optionally, in **Index options**, select:
  - **Unique values** to create a unique index.

- **Clustered index** to create a clustered index. If a clustered index already exists, this option is not available.
- **Ignore duplicate values** to control what happens when an INSERT statement inserts multiple, nonunique key values into an index. For more information, see <u>CREATE INDEX</u>.
- **Do not recompute statistics (not recommended)** to specify that index statistics are not automatically recomputed as the index is updated.
- **Filegroup** to specify the filegroup on which to create the index. Click the name of the filegroup.
- **Pad index** to leave space open on each interior node of the index. For more information, see <u>CREATE INDEX</u>.
- **Drop existing** to delete any existing index of the same name before creating the new index.
- **Fill factor** to specify how full SQL Server should make the leaf level of each index page during index creation. For more information, see <u>CREATE INDEX</u>.
- 10. Optionally, click **Edit SQL** to view and edit the Transact-SQL statement used to create the index.

### See Also

<u>Graphically Displaying the Execution Plan Using SQL Query Analyzer</u> <u>Placing Indexes on Filegroups</u>

### How to delete an index (Query Analyzer)

#### To delete an index

- 1. On the **Query** menu, click **Show Execution Plan**.
- 2. Execute the Transact-SQL script in the query pane.
- 3. In the result pane, click the **Execution Plan** tab.
- 4. Right-click the icon of the physical operator that represents the table containing the index that needs to be deleted, and then click **Manage Indexes**.
- 5. Optionally, in **Database** and **Table**, select the name of a different database and table.
- 6. In **Existing indexes**, click the name of the index to delete, and then click **Delete**.
- 7. Confirm the deletion.

### See Also

Graphically Displaying the Execution Plan Using SQL Query Analyzer

# Replication

Microsoft® SQL Server<sup>™</sup> 2000 replication is the process of copying and distributing data and database objects from one database to another and then synchronizing between databases for consistency.

Using replication, you can distribute data to different locations, to remote or mobile users over a local area network, using a dial-up connection, and over the Internet. Replication also allows you to enhance application performance, physically separate data based on how it is used (for example, to separate online transaction processing (OLTP) and decision support systems), or distribute database processing across multiple servers.

# **Replication Types (Enterprise Manager)**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 provides the following types of replication that you can use in your distributed applications:

- Snapshot replication
- Transactional replication
- Merge replication

Each type provides different capabilities depending on your application and different levels of ACID properties of transactions and site autonomy. For example, merge replication allows users to work and update data autonomously, although ACID properties are not assured. Instead, when servers are reconnected, all sites in the replication topology converge to the same data values. Transactional replication maintains transactional consistency, but Subscriber sites are not as autonomous as they are in merge replication because Publishers and Subscribers generally must be connected reliably and continuously for updates to be propagated to Subscribers.

It is common for the same application to use multiple replication types and options. Some of the data in the application may not require any updates at Subscribers, some sets of data may require updates infrequently, with updates made at only one or a few servers, while other sets of data may need to be updated daily at multiple servers.

Which type of replication you choose for your application depends on your requirements based on distributed data factors, whether or not data will need to be updated at the Subscriber, your replication environment, and the needs and requirements of the data that will be replicated. For more information, see <u>Planning for Replication</u>.

Each type of replication begins with generating and applying the snapshot at the Subscriber, so it is important to understand snapshot replication in addition to any other type of replication and options you choose.

# How to enable activation of the Interactive Resolver (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, in the Create Publication Wizard, on the Specify Articles page, click the table you want to publish, and then click its properties (...) button.
- 2. On the **Resolver** tab, click **Allow Subscribers to resolve conflicts interactively during on-demand synchronizations**.
- 3. In SQL Server Enterprise Manager, when creating a subscription in the Pull Subscription Wizard, select the option allowing the subscriber to resolve conflicts interactively during on-demand synchronizations.

Alternatively, you can set this option after you have created a pull subscription, on the **Synchronization** tab of the **Properties** dialog box for the subscription.

# To activate the Interactive Resolver during a merge synchronization (Windows Synchronization Manager)

- 1. On the Windows **Start** menu, point to **Programs**, point to **Accessories**, and then click **Synchronize**.
- 2. In the **Items to Synchronize** dialog box, click the subscription you want to synchronize, click **Properties**, and then click the **Other** tab.
- 3. Under **Conflict resolution mode**, click **Interactively resolve conflicts**, and then click **OK**.
- 4. Repeat Steps 1 through 3 for each subscription you will be synchronizing and using the Interactive Resolver.
- 5. Click **Synchronize**.

# How to set row- or column-level tracking for an article (Enterprise Manager)

- 1. In the Create Publication Wizard, on the **Specify Articles** page, click the **Table** you plan to use as an article in your merge publication.
- 2. Click the properties (...) button for the selected table.
- 3. On the Properties page for the article, on the **General** tab, under **When merging changes from different sources**, click **Treat changes to the same row as a conflict** for row-level tracking, or click **Treat changes to the same column as a conflict (changes to different columns in the same row will be merged)** for column-level tracking.

### How to choose a resolver (Enterprise Manager)

- 1. In the Create Publication Wizard, on the Specify Articles page, click the **Table** you plan to use as an article in your merge publication (if it is not already configured as an active article).
- 2. Click the properties (...) button for the selected table.
- 3. On the **Properties** page for the article, on the **Resolver** tab, click **Use the default resolver** to enable the default resolver for the article.
- 4. If you want to use a custom resolver with the article, click **Use this custom resolver**, and then in the list, click the desired resolver. If you want to use a custom stored procedure resolver, click **Microsoft SQL Server Stored Procedure Resolver**.
- 5. If you will be using a custom stored procedure resolver, press the TAB key to get to the **Information for the custom resolver** box, and then type the name of the stored procedure. If you are using a COM custom resolver, use the information box only if you need to enter any additional information required by the custom resolver (such as an input parameter).

# **Replication Tools (Enterprise Manager)**

Microsoft® SQL Server<sup>™</sup> 2000 provides several methods for implementing and administering replication, including SQL Server Enterprise Manager, programming interfaces, and other Microsoft Windows® components.

SQL Server Enterprise Manager includes a graphical organization of replication objects, several wizards, and dialog boxes you can use to simplify configuration and administration of replication. SQL Server Enterprise Manager allows you to view and modify the properties of replication configuration and monitor and troubleshoot replication activity.

You can also implement, monitor and maintain replication using programming interfaces such as Microsoft ActiveX<sup>®</sup> controls for replication, SQL-DMO, and scripting of Transact-SQL system stored procedures.

Components such as Windows Synchronization Manager and Microsoft Windows 2000 Active Directory<sup>™</sup> Services enable you to synchronize data, subscribe to publications, and organize and access replication objects from within Windows applications.

# How to open Publisher and Distributor properties (Enterprise Manager)

1. In SQL Server Enterprise Manager, expand the **Replication** folder, and then right-click the **Publications** folder.

### 2. Click **Configure Publishing**, **Subscribers**, and **Distribution**.

The Publisher and Distributor properties dialog box is titled with Publisher and Distributor Properties and the name of the Distributor.

# How to open publication properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** folder, and then expand the **Publications** folder.
- 2. Right-click a publication, and then click **Properties**.

If the publication has subscriptions, you will not be able to modify some properties, and a dialog box will notify you of this. Publication properties is titled with Publication Properties and the name of the publication.

# How to open push subscription properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** folder, and then expand the **Publications** folder.
- 2. Click a publication, right-click a push subscription, and then click **Properties**.

# How to open pull subscription properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** folder, and then click the **Subscriptions** folder.
- 2. Right-click a pull subscription, and then click **Properties**.

### How to open agent properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand Replication Monitor, and then expand the **Agents** folder.
- 2. Click the folder for the agent you want to see (for example, Snapshot Agents), right-click an agent, and then click **Agent Properties**.

The agent properties dialog box will be titled with the name of the Distributor, the published database, the publication, and job number.

# How to open Windows Synchronization Manager

**Note** Windows Synchronization Manager is installed automatically with Microsoft® Windows® 2000 and anywhere Microsoft Internet Explorer 5.0 or later is installed.

• On the Windows **Start** menu, click **Programs**, click **Accessories**, and then click **Synchronize**.

# **Implementing Replication (Enterprise Manager)**

Whether you are using snapshot replication, transactional replication, or merge replication, the following stages will help you implement replication.

Stage	Tasks
Configuring Replication	Identify the Publisher, Distributor, and Subscribers in your topology. Use SQL Server Enterprise Manager, SQL-DMO, scripts, or Transact-SQL system stored procedures to configure the Publisher, create a distribution database, and enable Subscribers.
Publishing Data and Database Objects	Create the publication and define the data and database object articles in the publication, and apply any necessary filters to data that will be published.
Subscribing to Publications	Create push, pull, or anonymous subscriptions to indicate what publications need to be propagated to individual Subscribers and when.
Applying the Initial Snapshot	Indicate where to save snapshot files, whether they are compressed, and scripts to run before or after applying the initial snapshot. Specify to have the Snapshot Agent apply the snapshot at the Subscriber immediately after creating a subscription or at a specified time.
	Apply the snapshot manually by saving it to a network location or to removable media that can be transported to the Subscriber, and then applying the Snapshot files manually at the Subscriber.
Synchronizing Data	Synchronizing data occurs when the Snapshot, Log Reader, or Merge Agent runs and updates are propagated between Publisher and Subscribers.

For snapshot replication, the snapshot will be reapplied at the Subscriber.
For transactional replication, updates will be propagated to Subscribers.
If using updatable subscriptions with either snapshot replication or transactional replication, data will be propagated from the Subscriber to the Publisher and to other Subscribers.
For merge replication, data is synchronized during the merge process when data changes at all servers are converged and conflicts, if any, are detected and resolved.

# How to configure publishing and distribution (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand a SQL Server group, expand a server, right-click the **Replication** folder, and then click **Configure Publishing, Subscribers, and Distribution**.
- 2. Follow the wizard pages to select a Distributor, create the distribution database, and then on the Customize the Configuration page, either accept Publisher and Subscriber defaults, or select **Yes, let me set the distribution database properties, enable Publishers or set the publishing settings**.

This allows you to set distribution database properties, enable Publishers, enable publication databases, and enable Subscribers using the wizard. You can also configure these properties later in the Publisher and Distributor properties.

# How to modify Publisher and Distributor properties (Enterprise Manager)

### To add, modify, or remove a Publisher

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Configure Publishing, Subscribers, and Distribution**.
- 2. Select what you want to do with the Publisher.
  - To add a Publisher, click the **Publishers** tab, and then select a Publisher to enable.
  - To modify a Publisher, click the **Publishers** tab, click the Publisher to modify, and then click the properties button (...).
  - To remove a Publisher, click the **Publishers** tab, and then clear the box next to the Publisher name.

**Note** If the Publisher does not have items in the distribution database, a confirmation prompt will not appear.

#### To modify a Distributor or add or modify a distribution database

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Configure Publishing, Subscribers, and Distribution**.
- 2. Select what you want to do with the Distributor.
  - To modify Distributor properties, click the **Distribution** tab to change the password for connecting to the Distributor or set agent profiles.

- To add a distribution database, click the **Distributor** tab, click **New**, and then enter a database name and the location for the database and log files.
- To modify the distribution database, click the **Properties** button for the distribution database to change the transaction retention period or the history retention period.

# How to add, modify, or disable a Subscriber (Enterprise Manager)

### To enable or disable a Subscriber

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Configure Publishing, Subscribers, and Distribution**.
- 2. Select what you want to do with the Subscriber.
  - To enable a Subscriber, click the **Subscribers** tab, and then if the Subscriber is listed, select the Subscriber.
  - To enable a Subscriber if it is not listed, on the **Subscriber** tab, click **New Subscriber**, click the type of Subscriber to register, and then enter the server, ODBC data source, or OLE DB data source, and connection information.
  - To disable a Subscriber, click the **Subscribers** tab, and then clear the box next to the Subscriber.

#### To modify a Subscriber

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Configure Publishing, Subscribers, and Distribution**.
- 2. Click the **Subscribers** tab, click the Subscriber to modify, and then click the properties button (...).
- 3. On the **General** tab, change the Subscriber description or the security mode.

4. Click the **Schedules** tab to modify default scheduling options.

## How to disable publishing and distribution (Enterprise Manager)

#### To disable a Distributor

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Disable Publishing**.
- 2. Complete the steps in the wizard.

#### To delete a distribution database

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Distributor, right-click the **Replication** folder, and then click **Configure Publishing and Distribution.**
- 2. Click the **Distributor** tab, select the database to delete, and then click **Delete**.

# How to create publications and define articles (Enterprise Manager)

### To create publications

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, right-click the Publications folder, and then click **New Publication**.
- 2. On the Welcome to the Create Publication Wizard page, select **Show advanced options in this wizard** to enable updatable subscriptions or transformable subscriptions (options available with snapshot replication or transactional replication).
- 3. The wizard guides you through:
  - Choosing a publication database.
  - Using a publication template.
  - Selecting the type of publication.
  - Selecting updatable subscriptions or transformable subscriptions (snapshot replication or transactional replication).
  - Specifying Subscriber types.
  - Specifying data and database object articles to publish.
  - Selecting a publication name and description. Publication

names cannot contain these characters: /  $\setminus$  < >.

• Customizing the properties of the publication including filtering columns, filtering rows, enabling dynamic filters, validating subscription information, optimizing synchronization, allowing anonymous subscriptions, and setting the snapshot agent schedule.

## How to modify publications and articles (Enterprise Manager)

**Note** If subscriptions have been created to the publication, some properties are disabled and cannot be changed.

#### To view or modify publication properties

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, right-click a publication and choose **Properties**.
- 2. In the **Publication Properties** dialog box, click the **General** tab to view the publication name, database name, and type of publication.
- 3. You can also view or modify the publication description, the initial snapshot file format, or the synchronization time limit.
- 4. Select what you want to do on the **General** tab.
  - To add, remove, or change the properties of an article, click the **Articles** tab. To view or change the schema objects that are being published, click the properties button (...) for an article and click the **Snapshot** tab
  - To filter the columns in published tables, click **Filter Columns**.
  - To filter the rows in published tables, click **Filter Rows**.
  - To push, delete, reinitialize, or view the properties of subscriptions, click the **Subscriptions** tab.
  - For snapshot or transactional publications, click **Subscription**

**Options** to allow pull subscriptions, allow anonymous subscriptions, allow new subscriptions to be created by attaching a subscription database, and view whether the publication allows transformations on published data, immediate updating, or queued updating.

- For merge publications, click **Subscription Options** to allow pull subscriptions, allow anonymous subscriptions, allow new subscriptions to be created by attaching a subscription database, view if data conflicts are stored centrally at the Publisher, view if dynamic filtering is used, and if so if Subscriber information is validated, and view if synchronization is being optimized.
- For snapshot and transactional publications, click the **Updatable** tab to see if immediate updating or queued updating subscriptions are allowed, to enable conflicts to be reported centrally at the Publisher, to specify the conflict resolution policy, and if queued updating is allowed, to specify where to queue changes at the Subscriber.
- To modify snapshot format (SQL Server or character mode), specify scripts to run before and after the snapshot is applied, and for transactional replication to enable concurrent snapshot processing, click the **Snapshot** tab.
- To specify an alternate location to save the snapshot, compress the snapshot files, and specify File Transfer Protocol (FTP) information if the Subscriber will access the snapshot folder using FTP, click the **Snapshot Location** tab.
- To specify the logins that have access to the publication, click **Publication Access List**.

- To view the status of the Snapshot Agent, run the Agent, view Agent properties, or start the services required by the publication, click the **Status** tab.
- For merge publications, click the **Sync Partners** tab to enable Subscribers to synchronize with servers other than the original Publisher, and then select the servers that may serve as alternate partners for this publication.

#### To grant or revoke access to a publication

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the Publications folder, right-click the publication, and then click **Properties**.
- 2. To add or remove a login for access to the publication, click the **Publication Access List** tab, click **Add**, **Remove**, or **Remove All**.

**Note** If a remote Distributor is used, the new logins must exist in the publication access lists at both the Publisher and at the Distributor. If the pull subscription login is not in the publication access list, an error appears at the Subscriber.

#### To add or delete an article

**Note** After subscriptions are created, articles can be added to existing publications. Deleting articles from publications that have subscriptions is not allowed. To remove an article, you must first delete all subscriptions.

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the Publications folder, right-click a publication and choose **Properties**.
- 2. Select what you want to do with the article.
  - To see a list of all available objects, click the **Articles** tab, and then select **Show unpublished objects**.
  - To add an article, select the **Show** check box next to the

**Object Type** listed. Select the check box next to the article object to add to the publication, or select the **Publish All** check box next to the object type you want to publish.

- To delete an article, select the **Show** check box next to the **Object Type** listed. Clear the check box next to the article object to delete to the publication, or clear the **Publish All** check box next to the object type you want to exclude from the publication.
- To set article options, click the build button (...).

## How to delete publications and articles (Enterprise Manager)

### To delete a publication

• At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the Publications folder, right-click the publication, and then click **Delete**.

### To delete an article

**Note** Deleting articles from publications that have subscriptions is not allowed. To delete an article, you must first delete all subscriptions to the publication.

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the Publications folder, right-click the publication, and then click **Properties**.
- 2. Click the **Articles** tab, select an article to delete, and then clear the check box next to the article to delete.

## How to create a push subscription (Enterprise Manager)

#### To create a push subscription

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, right-click the publication for which you want the subscription, and then click **Push New Subscription**.
- 2. Complete the steps in the wizard.

## How to modify a push subscription (Enterprise Manager)

### To modify push subscription properties

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, click the publication that has the subscription you want to modify, right-click the push subscription for that publication, and then click **Properties**.
- 2. Select what you want to do with the subscription properties.
  - To view the selected subscription properties, click the **General** tab.
  - To specify where the Distribution Agent or Merge Agent should run, click the **Synchronization** tab.

## How to delete a push subscription (Enterprise Manager)

### To delete a push subscription

• At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, click the publication that has the subscription you want to delete, right-click the push subscription for that publication in SQL Server Enterprise Manager, and then click **Delete**.

## How to create a pull or anonymous subscription (Enterprise Manager)

### To create a pull or anonymous subscription

- 1. At the Subscriber, in SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, right-click the Subscriptions folder, and then click **New Pull Subscription**.
- 2. Follow the steps in the **Pull Subscription Wizard**.

If the publication allows anonymous subscriptions, the **Allow Anonymous Subscription** page will show in the Pull Subscription Wizard and you can specify the new subscription as anonymous.

## How to view or modify pull or anonymous subscriptions (Enterprise Manager)

• At the Subscriber, in SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, click the Subscriptions folder, right-click the subscription you want to modify in the right pane of SQL Server Enterprise Manager, and then click **Properties**.

# How to delete a pull or anonymous subscription (Enterprise Manager)

• At the Subscriber, in SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, click the Subscriptions folder, right-click the subscription you want to delete in the right pane of SQL Server Enterprise Manager, and then click **Delete**.

## How to create an anonymous subscription (Windows Synchronization Manager)

- 1. On the **Start** menu, point to **Programs**, point to **Accessories**, and then click **Synchronize**.
- 2. Click **To create a subscription: select this, then click Properties**, click **Properties**, and then select **By specifying the publication and subscription information manually**.
- 3. Enter the name for the subscription, the Subscriber name, subscription database name, Publisher name, publication database name, type of publication, and Distributor name.

## How to view or modify the default snapshot folder location (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand a server group, expand the Publisher, right-click the Replication folder, and then click **Configure Publishers, Subscribers, and Distribution**.
- 2. Click the **Publishers** tab, and then click the distribution database properties button (...) for a specific Publisher.
- 3. To modify the default snapshot folder location, click the properties button (...) and browse to set a new default location.

## How to specify alternate snapshot locations (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directory, select a publication, right-click the publication, and then click **Properties**.
- 2. On the **Snapshot Location** tab, select **Generate snapshots in the following location** option, and then type a Universal Naming Convention path or click the browse button (...) and browse for the location where you want to save snapshot files.
- 3. To use compression, select **Compress the snapshot files in this location**.
- 4. If FTP is being used to transfer snapshots, select **Subscribers can** access this folder using FTP (File Transfer Protocol).

## How to compress and deliver snapshot files (Enterprise Manager)

#### To compress snapshot files

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories, right-click a publication, and then click **Properties**.
- 2. On the **Snapshot Location** tab, select **Generate snapshots in the following location**, specify a location for the files, and then select **Compress the snapshot files in this location**.

#### To configure snapshot delivery on the Subscriber

- 1. In Microsoft SQL Server Enterprise Manager, expand the subscription database and the **Subscriptions** directory, right-click a subscription, and then click **Properties**.
- 2. On the **Snapshot File Location** tab, select **Get the snapshot from the following folder**.
- 3. Type the path or click the browse (...) button and browse to the directory where you want snapshot files to be placed.

## How to set the – UseInprocLoader property (Enterprise Manager)

- 1. On the server where the Distribution Agent or Merge Agent is running, expand the **Replication Monitor** node, click the **Distribution Agents** or **Merge Agents** folder, right-click the agent that will be applying the snapshot, and then click **Agent Properties**.
- 2. On the **Steps** tab, double-click the subscription agent step, and then add the **–UseInprocLoader** property in the **Command** text box.

## How to execute scripts before and after the snapshot is applied (Enterprise Manager)

- 1. At the Publisher, open SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories, right-click a publication, and then click **Properties**.
- 2. On the **Snapshot** tab, click the browse (...) button for either **Before applying the snapshot** or **After applying the snapshot**, and then select the script that you want to execute before or after synchronization.

## How to reinitialize a subscription (Enterprise Manager)

### To reinitialize a push subscription

- At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, and then click the publication for which subscriptions need to be reinitialized.
- Right-click the subscription you want to reinitialize, and then click **Reinitialize**.

#### To reinitialize a pull or anonymous subscription

- At the Subscriber, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, and then click the **Subscriptions** folder.
- Right-click the subscription you want to reinitialize, and then click **Reinitialize**.

## How to browse and copy snapshot files (Enterprise Manager)

### To use the Snapshot Explorer

• In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories, select a publication, right-click the publication, and then click **Explore the Latest Snapshot Folder**.

## How to synchronize a subscription (Enterprise Manager)

#### To synchronize a push subscription

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, expand the **Publications** folder, and then click the publication for which subscriptions need to be synchronized.
- 2. Right-click the subscription you want to synchronize, and then click **Start Synchronizing**.

#### To synchronize a pull or anonymous subscription

- 1. At the Subscriber, open SQL Server Enterprise Manager, expand a server group, expand the **Replication** folder, and then click the **Subscriptions** folder.
- 2. Right-click the subscription you want to synchronize, and then click **Start Synchronizing**.

# How to synchronize an anonymous subscription (Windows Synchronization Manager)

#### To synchronize an anonymous subscription

- 1. On the **Start** menu, point to **Programs**, point to **Accessories**, and then click **Synchronize**.
- 2. Click the subscription that you want to synchronize, and then click **Synchronize**.

## How to view and resolve merge synchronization conflicts (Enterprise Manager)

### To view and further resolve synchronization conflicts

- 1. Expand a server group, and then expand a server.
- 2. Expand **Databases**, and then expand the name of the database.
- 3. Expand **Publications**, right-click the publication, and then click **View Conflicts**.
- 4. In the **Publications in database** list, select the publication to view.
- 5. In the **Tables with conflicts** list, select the table of conflicts to view.

**Note** Be sure to connect to the correct server to view the conflicts. The location of the conflict table varies depending upon whether replication has been configured for centralized of decentralize logging of conflicts. If centralized, the conflict table is stored at the Publisher and you must connect to the Publisher to view the conflicts. If decentralized, the conflict table is stored at either the Publisher or Subscriber, depending upon which one lost the conflicts.

## How to script replication (Enterprise Manager)

- 1. At the Publisher, open SQL Server Enterprise Manager, expand a server group, right-click the **Replication** folder, and then click **Generate SQL Script**.
- 2. Select the replication component to script (Distributor properties, publications and push subscriptions, or pull subscriptions) and whether you want the script to enable or create the components or disable or drop the components.

## How to apply schema changes on publication databases (Enterprise Manager)

### To add columns to an article

- 1. In SQL Server Enterprise Manager, under **Replication**, expand **Publications** and then right-click the publication where you want to modify a schema.
- 2. Click Properties, click Filter Columns, and then click Add Column.
- 3. In the **Add Column to Replicated Table** dialog box, enter the name of the column and the SQL syntax that defines the column. In the SQL for the column definition, you must either specify a default value or allow NULL values.
- 4. For information about the syntax required to define the column, see the Transact-SQL ALTER TABLE statement.
- 5. In the **Add Column to Replicated Table** dialog box, select the publications to which you want to add the column.

#### To drop columns from an article

- 1. In SQL Server Enterprise Manager, expand **Replication**, expand **Publications**, and then right-click the publication where you want to modify a schema.
- 2. Click **Properties**, click **Filter Columns**, select a table in the **Tables in publication** list, select a column in the **Columns in selected table** list, and then click **Drop Column**.
- 3. If the column is constrained, you will be prompted; columns with

primary key or unique constraints, and **uniqueidentifier** columns cannot be dropped. If you attempt to drop one of those types of columns, an error message is displayed. For other constraints, a warning message is displayed; click **OK** to drop the column.

## How to specify FTP information (Enterprise Manager)

#### To set the snapshot folder as the FTP home directory

- 1. On the **Start** menu, point to **Programs**, point to **Microsoft Internet Server**, and then click **Internet Service Manager**.
- 2. Click the server name corresponding to the FTP service.
- 3. On the **Properties** menu, click **Service Properties**, and then on the **Directories** tab, click **Add**.
- 4. Enter the path to the FTP directory (for example, C:\Microsoft SQL Server\Mssql\Repldata\Ftp), and then click **Home Directory**.

#### To configure the FTP home directory as an FTP site

- 1. On the **Start** menu, point to **Programs**, point to **Microsoft SQL Server 2000**, and then click **Client Network Utility**.
- 2. On the **General** tab, ensure that TCP/IP appears in the **Enabled protocols by order** list. If TCP/IP appears in the **Disabled protocols** list, select it, and then click **Enable**.
- 3. In **Server alias**, enter the name of the server.
- 4. In **Computer name**, overwrite the existing name with the IP address.
- 5. Your system administrator can provide you with the correct IP address.
- 6. In **Port number**, overwrite the existing port number, if necessary.

## **Replication Options (Enterprise Manager)**

Replication Options allow you to configure replication in a manner best suited to your application and environment.

Option	Type of Replication	Benefits
Filtering Published Data	Snapshot Replication Transactional Replication Merge Replication	<ul> <li>Filters allow you to create vertical and/or horizontal partitions of data that can be published as part of replication. By distributing partitions of data to different Subscribers, you can: <ul> <li>Minimize the amount of data sent over the network.</li> </ul> </li> <li>Reduce the amount of storage space required at the Subscriber.</li> <li>Customize publications and applications based on individual Subscriber requirements.</li> <li>Reduce conflicts because the different data partitions can be sent to different Subscribers.</li> </ul>
Updatable Subscriptions (Immediate Updating, Queued Updating)	Snapshot Replication Transactional Replication	Immediate updating and queued updating options allow users to update data at the Subscriber and either propagate those updates to the Publisher immediately or store the updates in a queue. Updatable subscriptions are best for replication topologies where replicated data

		is mostly read, and occassionally updated at the Subscriber when Publisher, Distributor, and Subscriber are connected most of the time and when conflicts caused my multiple users updating the same data are infrequent.
Transforming Published Data	Snapshot Replication Transactional Replication	<ul> <li>You can leverage the data movement, transformation mapping and filtering capabilities of Data Transformation</li> <li>Services (DTS) during replication. With transformable subscriptions, you can: <ul> <li>Create custom partitions for snapshot and transactional publications.</li> </ul> </li> <li>Transform the data as it is being published with data type mappings (for example, integer to real data type), column manipulations (for example, concatenating first name and last name columns into one), string manipulations, and functions.</li> </ul>
Alternate Synchronization Partners	Merge Replication	Alternate synchronization partners allow Subscribers to merge publications to synchronize data with servers other than the Publisher at which the subscription originated. This allows the Subscriber to synchronize data when the original Publisher is unavailable, and is also useful for mobile Subscribers that may have access to a faster or more reliable network connection with an alternate server.
Optimizing	Merge	By optimizing synchronization during

Synchronization	Replication	merge replication, you can store more information at the Publisher instead of transferring that information over the network to the Subscriber. This improves synchronization performance over a slow network connection, but requires additional storage at the Publisher.
Databases	Snapshot Replication Transactional Replication Merge Replication	Attachable subscription databases allow you to transfer a database with replicated data and subscriptions from one Subscriber to another. After the database is attached to the new Subscriber, the database at the new Subscriber will automatically receive its own pull subscriptions to the publications at those Publishers. This saves you the time and effort of creating subscription databases and subscriptions at multiple Subscribers.

## How to filter publications horizontally using the Create Publication Wizard (Enterprise Manager)

- 1. In the Create Publication Wizard, on the Customize the Properties of the Publication page, select **Yes, I will define data filters, enable anonymous subscriptions, or customize other properties**.
- 2. On the Filter Data page, select **Horizontally, by filtering the rows of published data**.
- 3. On the **Filter Table Rows page**, click the Filter Clause (...) button next to the article you want to filter, and then in the **Specify Filter** dialog box, complete the WHERE clause with a condition for the filter.

**Note** This is not the page to enter join filters that cross tables based on relationships between tables. That page in the Create Publication Wizard is called Generate Filters Automatically.

## How to filter publications vertically using the Create Publication Wizard (Enterprise Manager)

**Note** If you are creating a merge publication for use with Subscribers running Microsoft® SQL Server<sup>™</sup> version 7.0, you will not be able to create a vertical filter and you will not see the Filter Columns page in the Create Publication Wizard.

- 1. In the Create Publication Wizard, on the Customize the Properties of the Publication page, select **Yes, I will define data filters, enable anonymous subscriptions, or customize other properties**.
- 2. On the Filter Data page, select **Vertically, by filtering the columns of published data**.
- 3. On the Filter Table Columns page, click the table in the publication for which you want to add a vertical filter, and then clear the columns you do not want included in the publication.

# How to filter publications vertically using publication properties (Enterprise Manager)

**Note** If you are viewing the properties of a merge publication on Microsoft® SQL Server<sup>™</sup> version 7.0, you will not see the Filter Columns page.

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** folders, right-click the publication for which you want to add a column filter, and then click **Properties**.
- 2. On the **Filter Columns** tab, click the table in the publication for which you want to add a column filter, and then clear the columns you do not want included in the publication.

### How to validate Subscriber information using the Create Publication Wizard (Enterprise Manager)

- 1. In the Create Publication Wizard, on the Customize the Properties of the Publication page, select **Yes, I will define data filters, enable anonymous subscriptions, or customize other properties**.
- 2. On the Filter Data page, select **Horizontally, by filtering the rows of published data**, and then on the Enable Dynamic Filters page, select **Yes, enable dynamic filters**.
- 3. On the Filter Table Rows page, click the Filter Clause (...) button next to the article you want to filter, and then in the **Specify Filter** dialog box, complete the WHERE clause with the function that will retrieve information at the Subscriber and filter the publication dynamically.
- 4. On the Validate Subscriber Information page, select **Yes, validate Subscriber information**, and then type in the functions used in dynamic filters for this publication.

## How to filter publications horizontally using publication properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** folders, right-click the publication for which you want to add a row filter, and then click **Properties**.
- 2. On the **Filter Rows** tab, click the Filter Clause (...) button next to the article you want to filter, and then in the **Specify Filter** dialog box, complete the WHERE clause with a condition for the filter.

### How to create a dynamic snapshot (Enterprise Manager)

#### To generate a dynamic snapshot

**Note** You must generate a regular snapshot to the dynamically filtered merge publication before creating a dynamic snapshot.

- 1. Create the merge publication with dynamic filters enabled and specified on any necessary articles.
- 2. Generate the regular snapshot by running the Snapshot Agent.
- 3. At the Publisher, in SQL Server Enterprise Manager, expand the **Replication** and **Publications** folders, right-click the dynamically filtered merge publication, and then click **Create Dynamic Snapshot Job**.
- 4. On the Specify Filter Criteria page, type in the system functions used in the dynamic filters of the publication (SUSER\_SNAME() or HOSTNAME()) and the value of the login for the Publisher.
- 5. On the Specify Snapshot File Location page, type the path to the folder where you want snapshot files saved or click the browse button (...) and browse for the folder location. Using the alternate snapshot location feature, you can specify the snapshot folder location on the network, on removable media or on an FTP server.
- 6. On the Set Job Schedule page, select **Using the following schedule**, and then select **Change** to specify a schedule for when the dynamic snapshot will be generated, or select **On demand only**. Select the **Create the first snapshot immediately** check box to generate the dynamic snapshot immediately.

- 7. On the Specify Job Name page, type in a name for this dynamic Snapshot Agent.
- 8. Run the dynamic snapshot agent job

#### To apply the dynamic snapshot

- At the Subscriber, create a pull subscription using the Pull Subscription Wizard. On the Snapshot File Location page, select Use the snapshot from files from the following folder, specify or browse for the location of the dynamic snapshot in the text box, and then select This is a snapshot for a dynamically filtered subscription.
- 2. Finish the steps in the Pull Subscription Wizard. Manually start the Merge Agent (using Replication Monitor at the Publisher or programmatically) when the snapshot is available to apply it at the Subscriber.

### How to filter with a user-defined function using the Create Publication Wizard (Enterprise Manager)

- 1. In the Create Publication Wizard, on the Customize the Properties of the Publication page, select **Yes, I will define data filters, enable anonymous Subscribers, or customize other properties**.
- 2. On the Filter Data page, select **Horizontally, by filtering the rows of published data**.
- 3. If you also selected **Vertically, by filtering the columns of published data** on the Filter Data page, on the Filter Table Columns page, select the columns you want to filter from the publication.
- 4. In the **Filter Table Rows** dialog box, click the properties button (...) next to the article that you want to filter, and then complete the WHERE clause using a user-defined function in the condition.

### How to filter with a user-defined function using publication properties (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, on the **Tools** menu, click **Replication**, and then click **Create and Manage Publications**.
- 2. Expand the database containing the publication to modify, click the publication to modify, and then click **Properties & Subscriptions**.
- 3. If there are Subscriptions for this publication, you will need to delete them before you can modify the publication filter.

# How to drop all subscriptions to a publication (Enterprise Manager)

- 1. On the **Subscriptions** tab, click the subscription name, and then click **Delete**.
- 2. Click **OK** to close the **Publication Properties** dialog box, and then reopen it in the **Create and Manage Publications** dialog box by clicking **Properties & Subscriptions**.
- 3. In the **Publication Properties** dialog box, click the **Filter Rows** tab, and then complete the WHERE clause using a user-defined function as part of the filter condition.

## How to install Message Queuing on the Distributor and Subscribers (Enterprise Manager)

- 1. In Control Panel, double-click **Add/Remove Programs**, click **Add/Remove Windows Components**, and then select **Message Queuing Services**.
- 2. Select Message Queuing Server.

### How to set the queued updating conflict resolution policy (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories.
- 2. Right-click the publication that allows queued updating, and then click **Properties**.
- 3. On the **Updatable** tab, under **Conflict resolution policy**, you can choose to keep the change made at the Publisher, keep the change made at the Subscriber, or reinitialize the subscription.

# How to allow decentralized conflict reporting (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories.
- Right-click the publication that allows queued updating, and then click **Properties**. On the **Updatable** tab, under **Data Conflicts**, you can clear **Report conflicting data changes at the Publisher only**. Conflicting data changes will be reported at both the Publisher and Subscriber.

### How to view conflicts (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories.
- 2. Right-click the publication that allows queued updating, and then select **View Conflicts**.

### How to enable immediate updating with queued updating as a failover (Enterprise Manager)

- 1. In the Create Publication Wizard, on the Welcome page, select **Show advanced options in this wizard**.
- 2. Select either **Snapshot publication** or **Transactional publication**, and then on the Updatable Subscriptions page, select both **Immediate updating** and **Queued updating**.
- 3. When you create a subscription using either the Pull Subscription Wizard or Push Subscription Wizard, on the Updatable Subscriptions page, select **Immediate updating with queued updating as a standby in case of failure**.

## How to switch from immediate updating to queued updating as a failover (Enterprise Manager)

- 1. For push subscriptions: expand the **Replication** and **Subscriptions** directories, right-click the subscription, click **Set Update Method**, and then select either **Immediate Updating** or **Queued Updating**.
- 2. For pull subscriptions: on the Subscriber, expand the **Pull Subscriptions** directory, right-click the subscription, click **Properties**, and then on the **Synchronization** tab, select either **Immediate Updating** or **Queued Updating**.

# How to switch from immediate updating to queued updating as a failover (Transact-SQL)

• Use the **sp\_setreplfailovermode** stored procedure and set the following parameters.

Parameter	Description
@publisher	Name of the Publisher.
@publisher_db	Name of the publication database.
@publication	Name of the publication.
@failover_mode	Can be 'immediate', or 'queued'.

### How to switch from immediate updating to queued updating as a failover (Windows Synchronization Manager)

- 1. On the **Start Menu**, point to **Programs**, point to **Accessories**, and then click **Synchronize**.
- Double-click the subscription, and then in the Properties dialog box, on the Other tab, select either Immediate Updating or Queued Updating.

Pull subscriptions created using on-demand synchronization are automatically added to Windows Synchronization Manager. You can add pull subscriptions that are not using on-demand synchronization to Windows Synchronization Manager by opening the subscription properties, and then on the **Synchronization** tab, selecting **Enable this subscription to be synchronized using the Windows Synchronization Manager**.

#### How to create a transformable subscription (Enterprise Manager)

- 1. Create a publication enabled for transformable subscriptions using the Create Publication Wizard, and on the Create Publication Wizard Welcome page, click **Show Advanced Options**, click **Next**, and then click a database in the list.
- 2. On the Choose Publication Type page, click either **Snapshot Publication** or **Transactional Publication**.
- 3. On the Updatable Subscriptions page, do not select **Immediate Updating** or **Queued Updating** because transformable subscriptions will not also be available.
- 4. On the **Transform Published Data** page, click **Yes**, and then continue creating the publication.
- 5. Build the replication DTS package using the Transform Published Data Wizard.
  - a. In SQL Server Enterprise Manager, right-click the publication enabled for transformable subscriptions, and then click Properties.
  - b. In the **Properties** dialog box for the publication, click **Subscriptions**, and then click **Transformations**.
  - c. In the Transform Published Data Wizard, click Next until the Choose a Destination page is displayed. Select a provider to connect to the Subscriber (the Microsoft® SQL Server<sup>™</sup> 2000 Replication OLE DB Provider for DTS is used only for the source connection from the package to the Distributor),

and then complete the rest of the connection information. On the Define Transformations page, click the transform (...) button for a published table on which to define a transformation.

- d. On the Column Mappings and Transformations page, click the Column Mappings tab, and then click one of the following: Keep the existing table unchanged, DROP the existing table and re-create it or Delete all data in the existing table. If you want to partition data vertically, in the list, select the Destination columns you want to include.
- e. If you want to partition published data horizontally or map transformations on the published partition, click the Transformations tab for the published table. Click Transform data using the following script. Choose the type of language for the script from the drop down, edit directly in the script window or click Load file to load a script. Repeat this step for all tables in the publication for which you want to add transformations.
- f. On the Save DTS Package page, enter a **Name**, enter a **Description**, and then either enter optional password information for the package or use the supplied default values. Continue until the package is created successfully. If you want to use this package for a push subscription, save the package at the Distributor (default). If using the package for a pull subscription, save it at the Subscriber.
- 6. This set of steps is used to create a transformable subscription.
  - a. Open either the Push Subscription Wizard or the Pull Subscription Wizard.
  - b. On the Specify DTS Package page, click the DTS package you want to use with the subscription, optionally enter a

package password, click **Next**, and then continue through the wizard until you have created the subscription successfully.

### How to enable Subscribers to synchronize with alternate synchronization partners (Enterprise Manager)

- 1. In SQL Server Enterprise Manager, expand the **Replication** and **Publications** directories, right-click a Publication, and then click **Properties**.
- 2. Click the **Sync Partners** tab, and then enable **Allow Subscribers to synchronize with other partners than the Publisher from which the subscription was created**.
- 3. Enable the Publishers and Subscribers that can be alternate Publishers for Subscribers to this publication.

### How to enable a Subscriber at an alternate synchronization partner (for named subscriptions) (Enterprise Manager)

- On the Publisher, right-click the **Replication** directory in SQL Server Enterprise Manager, and then click **Configure Publishing**, **Subscribers, and Distribution**.
- 2. On the **Subscribers** tab, select the box next to the Subscriber you want to enable or click **New Subscriber** and register the Subscriber.

### How to synchronize with alternate synchronization partners (Windows Synchronization Manager)

- 1. On the **Start Menu**, point to **Programs**, **Accessories**, and then click **Synchronize**.
- 2. Select the publication you want to synchronize, click **Properties**, and then on the **Identity** tab, select the Publisher with which you want to synchronize.
- 3. Click **OK**, and then click **Synchronize**.

### How to synchronize pull subscriptions with alternate synchronization partners (Enterprise Manager)

- On the Subscriber, expand the **Replication** directory, click the Subscriptions directory, right-click a pull subscription, and then click **Properties**.
- 2. On the **Synchronization** tab, click **Merge Agent Properties**.
- 3. On the **Steps** tab, double-click the subscription agent step, and then add the **–SyncToAlternate** switch in the **Command** text box.

## How to synchronize push subscriptions with alternate synchronization partners (Enterprise Manager)

- 1. On the Subscriber, expand the **Replication Monitor** and **Agents** directories, select the **Merge Agents** directory, right-click the publication, and then click **Agent Properties**.
- 2. On the **Steps** tab, double-click the subscription agent step, and add the **–SyncToAlternate** switch in the **Command** text box.

# How to minimize the amount of data sent over the network during merge replication (Transact-SQL)

• Execute the **sp\_addmergepublication** system stored procedure and set the **@keep\_partition\_changes** parameter to '**true**'.

### How to configure a publication to allow copying of subscription databases (Enterprise Manager)

- 1. Create the publication using the Create Publication Wizard.
- 2. In SQL Server Enterprise Manager, expand the **Replication** and **Publication** directories, and then right-click the publication that you want to enable for new subscriptions.
- 3. For snapshot replication and transactional replication, click the **Subscription Options** tab in the Publication Properties, select **Use a Distribution Agent that is independent of other publications from this database**, and then select **Snapshot files are always available to immediately initialize new subscriptions**. You do not need to do anything for this step if you are using merge replication.
- 4. Select Allow new subscriptions to be created by attaching a copy of a subscription database.

#### How to copy a subscription database (Enterprise Manager)

- 1. On the Subscriber, in SQL Server Enterprise Manager, expand the **Replication** and **Subscriptions** directories, right-click the pull subscription that has a subscription database you want to copy, and then click **Copy Subscription Database**.
- 2. In the **Copy Subscription Database** dialog box, browse to the directory or drive where you want to save a copy of the subscription database.
- 3. The location can be on the network, using removable media (such as CD-ROMs or tape devices) or on a File Transfer Protocol (FTP) site.
- 4. In the **File Name** box, type a name for the subscription database file (the file will have the extension .msf).

# How to enable a Subscriber to receive published data (Enterprise Manager)

- 1. On the Publisher, in SQL Server Enterprise Manager, right-click the **Replication** directory, and then click **Configure Publishing**, **Subscribers, and Distribution**.
- 2. On the **Subscribers** tab, select the box next to the Subscriber you want to enable or click **New Subscriber** and register the Subscriber.

### How to attach a subscription database with named subscriptions (Enterprise Manager)

- 1. On the new Subscriber, in SQL Server Enterprise Manager, expand the **Replication** directory.
- 2. Right-click the **Subscriptions** directory, and then click **Attach Subscription Database**.
- 3. Either type the Universal Naming Convention (UNC) path or click the browse (...) button and browse for the location of the .msf file.
- 4. In the **Name of database to create** box, type a name for the database.

### How to attach a subscription database with anonymous subscriptions (Enterprise Manager)

- 1. On the new Subscriber, in SQL Server Enterprise Manager, expand the **Replication** directory.
- 2. Right-click the **Subscriptions** directory, and then click **Attach Subscription Database**.
- 3. Either type the UNC path or click the browse (...) button and browse for the location of the .msf file.
- 4. In the **Attach as database** box, type a name for the database.

### Administering and Monitoring Replication (Enterprise Manager)

SQL Server replication provides tools to administer and monitor replication agents and replication alerts and replication processes so that you can ensure that replication is meeting the needs of your applications and your organization.

Monitoring replication will help you:

- Set the profiles, schedules and notifications for replication agents.
- Troubleshoot agent activity including verifying when agents last ran, monitoring agent activity.
- Troubleshoot agent errors.
- Ensure that data values are the same at the Publisher and at Subscribers.

# How to change replication monitoring properties (Enterprise Manager)

#### To change replication monitoring properties

- 1. Expand a server group, and then expand the Distributor.
- 2. Right-click Replication Monitor, and then click **Refresh Rate and Settings**.

#### To select columns for monitoring views

- 1. Expand a server group; then expand the Distributor.
- 2. Right-click Replication Monitor; then click **Select Columns**.

### How to monitor replication agent history (Enterprise Manager) To monitor replication agent history

- 1. Expand a server group, and then expand the Distributor.
- 2. Expand Replication Monitor, and then click Agents.
- 3. Click the agent to monitor, right-click a row in the details pane, and then click **Agent History**.

### How to configure DCOM to run the Distribution Agent remotely

#### To configure DCOM to run the Distribution Agent remotely

**Note** To run DCOM configuration, on Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000, run Dcomcnfg.exe located at \Winnt\System32. On Microsoft Windows 98, run Dcomcnfg.exe located at \Windows\System.

- 1. On the computer where you want the agent to run, on to the **Start** menu, click **Run**, type **dcomcnfg**, and then click **OK**.
- 2. On the **Applications** tab, select **Microsoft SQL Server Replication Remote Dist Agent 8.0**, and then click **Properties**.
- 3. On the **Security** tab, select **Use custom launch permissions**, and then click **Edit**.
- 4. In the Registry Value Permissions window, add the account used to run SQL Server Agent on the Distributor (for push subscriptions), or SQL Server Agent on the Subscriber (for pull subscriptions), and then click **OK**.
- 5. Click the **Identity** tab, select **This user**, and then type the user account used by SQL Server Agent on the Distributor (for push subscriptions), or SQL Server Agent on the Subscriber (for pull subscriptions).

### How to configure DCOM to run the Merge Agent remotely

#### To configure DCOM to run the Merge Agent remotely

**Note** To run DCOM configuration, on Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000, run Ecomenfg.exe located at \Winnt\System32. On Microsoft Windows 98, run Decomenfg.exe located at \Windows\System.

- 1. On the **Start** menu, click **Run**, enter **dcomcnfg**, and then click **OK**.
- 2. On the **Applications** tab, select **Microsoft SQL Server Replication Remote Merge Agent 8.0**, and then click **Properties**.
- 3. On the **Security** tab, select **Use custom launch permissions**, and then click **Edit**.
- 4. In the Registry Value Permissions window, add the account used to run SQL Server Agent on the Distributor (for push subscriptions), or SQL Server Agent on the Subscriber (for pull subscriptions), and then click **OK**.
- 5. Click the **Identity** tab, select **This user**, and then enter the same user account used by SQL Server Agent on the Distributor (for push subscriptions), or SQL Server Agent on the Subscriber (for pull subscriptions).

## How to enable a push subscription to use remote agent activation

#### To configure a push subscription to use remote agent activation

- 1. In the Microsoft® Management Console on the Distributor, expand **Microsoft SQL Servers 2000**, expand **SQL Server Group**, select the Distributor, and then expand the **Replication** directory.
- 2. At the Publisher, expand the **Publications** directory, right-click a publication, and then select **Push New Subscription**.
- 3. In either the Set Distribution Agent Location window for Transactional Replication, or the Set Merge Agent Location window for Merge Replication, select **Run the Agent at the Subscriber**.
- 4. To verify that the agent can be run remotely, click **Verify Subscriber**.
- The Distributor connects to the Subscriber and starts the Distribution Agent or the Merge Agent using DCOM. If the connection is successful, you will receive the message, **The Subscriber** 'SubscriberName' is prepared to run the offload agent.

**IMPORTANT** After you specify where the agent should run when creating the subscription, synchronization may fail if you specified that the subscription should be automatically synchronized and you haven't configured DCOM for the remote agent activation.

## How to enable a pull subscription to use remote agent activation

### To configure a pull subscription to use remote agent activation

- 1. At the Subscriber, in SQL Server Enterprise Manager, expand the **Replication** folder, right-click the **Subscriptions** folder, and then click **New Pull Subscription.**
- 2. Follow the steps in the wizard to create a new pull subscription.
- 3. After the pull subscription is created, right-click on it, and then select **Properties**.
- 4. Click the **Synchronization** tab, and then select **Run the agent at the Distributor** to offload agent processing from the Subscriber.

**IMPORTANT** After you specify where the agent should run when creating the subscription, synchronization may fail if you specified that the subscription should be automatically synchronized and you haven't configured DCOM for the remote agent activation.

# How to configure an existing subscription to use remote agent activation

To configure an existing subscription to use remote agent activation

- 1. Right-click a publication, and then select **Properties**.
- 2. Click the **Subscriptions** tab, and then select **Properties**.
- 3. Click the **Synchronization** tab, and then select **Run the distribution agent at the Subscriber** for push subscriptions or **Run the distribution agent at the Distributor** for pull subscriptions.

# How to monitor replication agent performance (Enterprise Manager)

## To monitor replication agent performance

- 1. Expand a server group; then expand a server.
- 2. Right-click Replication Monitor; then click **Performance Monitor**.
- 3. On the **Edit** menu, click **Add To Chart**.
- 4. In the **Object** list, select the SQL Server replication object to monitor.
- 5. In the **Counter** list, select the counter to use.

#### To monitor replication agent session details

- 1. Expand a server group; then expand the Distributor.
- 2. Expand Replication Monitor; then click Agents.
- 3. Click the agent to monitor.
- 4. Right-click a row in the details pane; then click **Agent History**.
- 5. Select a session in the session list; then click **Session Details**.

## How to create a replication agent profile (Enterprise Manager)

### To create a replication agent profile

- 1. Expand a server group, and then expand the Distributor.
- 2. On the **Tools** menu, point to **Replication**, and then click **Configure Publishing**, **Subscribers**, and **Distribution**.
- 3. Click **Agent Profiles**, click the tab for the type of agent to get a new profile, and then click **Copy Selected Profile**.
- 4. Enter the name and optional description of the new profile, click the parameters you want to change, and then enter the new value.

#### To set the default profile for a type of replication agent

- 1. Expand a server group, and then click the Distributor name.
- 2. On the **Tools** menu, point to **Replication**, and then click **Configure Publishing**, **Subscribers**, **and Distribution**.
- 3. Click the **Distributor**, click **Agent Profiles**, and then click the tab for the type of replication agent.
- 4. Select the **Default** column next to the profile to be used as the default.
- 5. Select **Change all existing** *type* **Agents to use the selected profile** to apply the new default to all existing *type* agents. Clearing this option will apply the new default only to new agents created from this point forward.

#### To view or modify a replication agent profile

- 1. Expand a server group, expand a server, and then expand Replication Monitor.
- 2. Expand **Publishers**, expand the Publisher name where the profile is to be modified, and then click the publication.
- 3. In the details pane, right-click the agent or subscription, and then click **Agent Profiles**.
- 4. Select an agent profile, click **View Details**, and then enter the value of the parameter you want to change. If you created the agent profile, you can click **Modify** to modify the parameters for the agent.

**Note** You cannot delete the system profiles, and you cannot delete a profile if it is being used by any agent.

### To delete a replication agent profile

- 1. Expand a server group, and then click the Distributor name.
- 2. On the **Tools** menu, point to **Replication**, and then click **Configure Publishing**, **Subscribers**, **and Distribution**.
- 3. Click the **Distributor** tab, click **Agent Profiles**, and then click the tab for the type of replication agent.
- 4. Select the profile to be deleted, and then click **Delete Profile**.

# **Replication and Heterogeneous Data Sources** (Enterprise Manager)

Microsoft® SQL Server<sup>™</sup> 2000 offers the ability to replicate data to any heterogeneous data source that provides a 32-bit ODBC or OLE DB driver on Microsoft Windows® 2000, Microsoft Windows NT® Server 4.0, or Windows 98 operating systems. Additionally, SQL Server 2000 can receive copies of data replicated from Microsoft Access, Microsoft Exchange, Oracle, DB2 Universal, DB2/MVS, and DB2 AS400.

## **Heterogeneous Subscribers**

Publishing to heterogeneous data sources allows corporations that have acquired different databases to continue providing SQL Server 2000 to individuals or offices using those databases.

The simplest way to publish data to a heterogeneous data source is by using ODBC and creating a push subscription from the Publisher to the ODBC Subscriber.

## **Heterogeneous** Publishers

SQL Server 2000 can subscribe to snapshot or transactional data replicated from Oracle, DB2, Access, and other data sources. This allows companies that are planning to deploy large databases or a data warehouse with SQL Server, or Internet and intranet applications, to gain access to various sources of data. That data can then be consolidated in SQL Server 2000 using replication, and placed into a data mart, data warehouse, or multidimensional database designed for SQL Server Analysis Services.

To implement snapshot or transactional replication published by heterogeneous data sources to your SQL Server 2000 applications, configure SQL Server with third-party software or using applications built with SQL-DMO and the Replication Distributor Interface.

For more information, see <u>Programming Replication from Heterogeneous Data</u> <u>Sources</u>.

# How to publish to heterogeneous Subscribers (Enterprise Manager)

- 1. Expand a server group; then expand the Publisher.
- 2. On the Tools menu, point to Replication, and then click Create and Manage Publications.
- 3. In the **Databases and Publications** list, click the database from which to create a publication, and then click **Create Publication**.
- 4. When prompted to specify subscriber types that will use the publication, select **One or more Subscribers will not be SQL Servers**.
- 5. Complete the steps in the wizard.
- 6. On the Tools menu, point to Replication, and then click Configuring Publishing, Subscribers, and Distribution.
- 7. Click the **Subscribers** tab.
- 8. Click **New Subscriber**; then click the type of data source.
- 9. Select the heterogeneous Subscriber; then specify the login information, if required.

# How to enable a Jet 4.0 database as a Subscriber (Enterprise Manager)

- 1. Expand a server group; then expand the Publisher of the publication to which the Jet Subscribers will subscribe.
- 2. On the **Tools** menu, point to **Replication**, and then click **Configuring Publishing**, **Subscribers**, **and Distribution**.
- 3. On the **Subscribers** tab, click **New Subscriber**.
- 4. Select Microsoft Jet 4.0 database (Microsoft Access).
- 5. Select the new Subscriber from the list of Microsoft® Jet 4.0 databases shown. Enter the login name and, optionally, the password for the Microsoft Jet database. The Microsoft Jet database does not need to exist. If the database is new or unsecured, you must enter **admin** as the login name.
- 6. If the database is not listed, click **Add**. In **Linked server name**, enter a name for the linked server (for example, enter the name of the Jet database).
- 7. In **Database path**, enter the path and file name to the database. If the database is located on the same server as the Distributor, you can use local drive letters in the path. If the database is located on a different server than the Distributor, enter a UNC path. If the database does not exist, it will be created automatically when the subscription is initialized.

# How to create a publication for a Jet 4.0 Subscriber (Enterprise Manager)

- 1. Expand a server group; then expand the Publisher.
- 2. On the **Tools** menu, point to **Replication**, and then click **Create and Manage Publications**.
- 3. In the **Databases and Publications** list, click the database from which to create a publication list, and then click **Create Publication**.
- 4. Follow the steps in the wizard.
- 5. If you are creating a transactional publication, select **One or more Subscribers will not be a server running SQL Server** when asked **What type of Subscribers will subscribe to this publication?**
- 6. If you are creating a merge publication, select Some Subscribers will be Microsoft Jet 4.0 databases when asked What type of Subscribers will subscribe to this publication?
- 7. Follow the remaining steps in the wizard.

# How to add a push subscription to a Jet 4.0 Subscriber (Enterprise Manager)

- 1. Expand a server group; then expand the Publisher.
- 2. On the **Tools** menu, point to **Replication**, and then click **Push Subscriptions to Others**.
- 3. Expand the database containing the publication to which a push subscription will be added.
- 4. Click the publication; then click **Push New Subscription**.
- 5. Follow the steps in the wizard.
- 6. If the subscription is to a merge publication, when asked to set the subscription priority, click the priority that corresponds to the type of subscription known to Microsoft® Access. If you want the subscription to be an Access Local or Anonymous replica, select **Use the priority setting of the Publisher from which this subscription is created**. If you want the subscription to be an Access Global replica, select **Use the following priority to resolve the conflict** and set the desired priority.
- 7. Follow the remaining steps in the wizard.

# **Replication Security (Enterprise Manager)**

Replication security is an important part of the design and implementation of your distributed application. Replication applies the data changes made elsewhere on the network to the database at your server and vice-versa.

The decentralized availability of replicated data increases the complexity of managing or restricting access to that data. Microsoft® SQL Server<sup>™</sup> 2000 replication uses a combination of security mechanisms to protect the data and business logic in your application:

• Role requirements

By mapping user logins to specific SQL Server 2000 roles, SQL Server 2000 allows users to perform only those replication and database activities authorized for that role. Replication grants certain permission to the **sysadmin** fixed server role, the **db\_owner** fixed database role, the current login, and the **public** role. For example, only members of the **sysadmin** server role can configure replication.

• Distributor administrative link security

SQL Server 2000 provides a secure administrative link between the Distributor and a remote Publisher. Publishers can be treated as trusted or nontrusted.

• Snapshot folder security

The operating system or FTP service prevents users from accessing specific files on the server. The user must have a valid login to read or write the files used in the replication process.

• Registered subscribers

SQL Server 2000 allows you to limit access to publications to either registered Subscribers that are well-known to the Publisher, anonymous, or Subscribers that have logins in the publication access list. SQL Server 2000 uses linked server definitions for heterogeneous Subscribers to secure the replication of data with heterogeneous data sources. • Publication access lists

By supporting publication access lists (PAL) on each server, SQL Server 2000 allows you to determine which logins have access to publications. SQL Server 2000 creates the PAL with default logins, but you can add or delete logins from the list.

• Agent login security

By supporting agent login security, SQL Server 2000 requires each user to supply a valid login account to connect to the server. Replication agents are required to use valid logins when connecting to Publishers, Distributors, and Subscribers. However, agents also can use different logins and security modes when connecting to different servers simultaneously.

• Immediate-updating Subscriber security

For immediate-updating Subscribers, SQL Server 2000 replication applies security mechanisms to the Publisher-RPC link and Publisher stored procedures.

When used together, these security mechanisms provide the highest safeguards for the data and business logic in your application.

## How to change the login property of a pull subscription

- 1. On the **Tools** menu, point to **Replication**, and then click **Pull Subscription To**.
- 2. Click **Properties**, and then on the **Synchronization** tab, click **Distribution Agent Properties**.
- 3. On the **Steps** tab, click **Edit**.
- 4. On the command line, edit the values for **SubscriberSecurityMode** and **SubscriberLogin**.

## How to add or change a password on a Distributor

### To add or change a password on a Distributor

- 1. Expand a server group; then click the Distributor name.
- 2. On the Tools menu, point to Replication, and then click Configure Publishing, Subscribers, and Distribution.
- 3. Click the **Distributor** tab.
- 4. Enter the new or changed password.

## To grant or revoke access to a publication

### To grant or revoke access to a publication

- 1. Expand a server group; then click the Publisher name.
- 2. On the Tools menu, point to Replication, and then click Create and Manage Publications.
- 3. Expand the database; then click the publication.
- 4. Click Properties & Subscriptions.
- 5. Click the Publication Access List tab.
- 6. Add or remove the login to access the publication.

**Note** If a remote Distributor is used, the new logins must exist in the publication access lists at the Publisher and the Distributor. If the pull subscription login is not in the publication access list, an error appears at the Subscriber.

# **Data Transformation Services**

This section contains procedures for:

- Using Data Transformation Services (DTS) tools.
- Using DTS connections, tasks, transformations, and workflow elements to build DTS packages.
- Managing packages.

# **DTS Tools**

This section contains procedures for using Data Transformation Services (DTS) tools.

## How to create a connection to Northwind in DTS Designer (Enterprise Manager)

### To create a connection to Northwind in DTS Designer

- 1. From the **Connection** toolbar, drag a Microsoft® OLE DB Provider for SQL Server connection onto the Data Transformation Services (DTS) Designer design sheet.
- 2. In the **New Connection** box, type **Cn1**.
- 3. In the **Data source** list, click **Microsoft SQL OLE DB Provider for SQL Server**.
- 4. In the **Server** list, click **local**.
- 5. Do one of the following:
  - Click Use Windows Authentication.

-or-

- Click **Use SQL Server Authentication**, and then enter a user name and password.
- 6. In the **Database** list, click **Northwind**.

## How to create a second connection to the Northwind database using DTS Designer (Enterprise Manager)

To create a second connection to the Northwind database using DTS Designer

- 1. From the **Connection** toolbar, drag a second Microsoft® OLE DB Provider for SQL Server connection onto the Data Transformation Services (DTS) Designer design sheet.
- 2. In the **New Connection** box, type **Cn2**.
- 3. Complete the remaining property selections as you did for the first connection.

# How to copy data from a Northwind table using DTS Designer (Enterprise Manager)

### To copy data from a Northwind table using DTS Designer

1. On the Data Transformation Services (DTS) Designer design sheet, CTRL-click **Cn1**, and then CTRL-click **Cn2**.

The order in which you CTRL-click is the order in which DTS Designer directs the data flow.

2. On the **Task** toolbar, click **Transform Data Task**.

An arrow appears pointing from **Cn1** to **Cn2**.

- 3. Right-click the **Transform Data** arrow, and then click **Properties**.
- 4. In the **Description** box, type **Copy Categories data**, and then in the **Table/View** list, click **[Northwind].[dbo].[Categories]**.
- 5. Click the **Destination** tab, and then click **Create**.

A new table is created to receive the copy of the source data.

6. In the **Create Destination Table** dialog box, in the **SQL statement** box, position the insertion point in the first line and edit the CREATE TABLE statement so it reads:

## CREATE TABLE [Categories2]

7. Click the **Transformations** tab and view the mappings between the source and destination columns in the two tables. Then click **OK** to exit the **Transform Data Task Properties** dialog box and save the settings.

**Note** In this example, you are copying data, but you also can use this dialog box to map transformations or manipulate the columns.

# How to configure an Execute SQL task to drop and re-create a destination table (Enterprise Manager)

To configure an Execute SQL Task to drop and re-create a destination table

- 1. From the **Task** toolbar, drag an Execute SQL task onto the Data Transformation Services (DTS) Designer design sheet.
- 2. In the **Description** box, type **Drop Dest Table**.

After you configure the task, that text will display on the design sheet, under the Execute SQL task icon.

3. In the **Existing connection** list, click **Cn2**.

In this example, you can use either connection because both **Cn1** and **Cn2** connect to the same database. However, it is better practice to use the destination connection.

4. In the **SQL statement** text box, type the following SQL code:

IF EXISTS (SELECT \* from sysobjects WHERE id = object\_id(N'[Northwind].[dbo].[Categories2]') OBJECTPROPERTY(id, N'IsUserTable') = 1) DROP Table [Northwind].[dbo].[Categories2] GO

```
CREATE TABLE [Northwind].[dbo].[Categories2]
(
[CategoryID] [int] IDENTITY (1,1) NOT NULL PRIMA
[CategoryName] [nvarchar] (15) NOT NULL,
[Description] [ntext],
[Picture] [image]
)
GO
```

This SQL code checks for the presence of the destination table. If the table does not exist, it is created. If the table exists, it is dropped and re-created. Without this package step, the same data from the source table is appended to the destination table every time the package is run.

5. Click **OK** to save the configuration settings and SQL code for the Execute SQL task.

# How to configure workflow in the Execute SQL task (Enterprise Manager)

## To configure workflow in the Execute SQL Task

- 1. On the Data Transformation Services (DTS) design sheet, CTRL-click **Drop Dest Table** (the Execute SQL task you created), and then CTRL-click **Cn1** (the first **Northwind** connection).
- 2. Click the **Workflow** menu, and then click **On success**.

A green striped arrow appears pointing from **Drop Dest Table** to **Cn1**. This arrow is a conditional precedence constraint. It directs the workflow so that the first task must execute successfully in order for the next task (the Transform Data task) to run.

# How to save the DTS package to a SQL Server msdb table (Enterprise Manager)

### To save the DTS package to a SQL Server msdb table

- 1. In Data Transformation Services (DTS) Designer, on the **Package** menu, click **Save**.
- 2. In the **Package name** box, type **Northwind Package**.
- 3. In the **Location** list, click **SQL Server**.

# How to access a DTS package template (Enterprise Manager)

### To access a DTS package template

- 1. In SQL Server Enterprise Manager, right-click **Data Transformation Services**, point to **All Tasks**, and then click **Open Template**.
- 2. In the **Select File** dialog box, double-click the template you want (.dtt file).

# How to create and save a DTS package template (Enterprise Manager)

### To create and save a DTS package template

1. In Data Transformation Services (DTS) Designer, right-click the design sheet of a DTS package, click **Disconnected Edit**, and stub out property values that will be entered later by template users. Repeat as needed.

You stub out a property value by replacing it with a label or instruction or by deleting the current value.

- 2. Right-click the design sheet, click **Add Text Annotation**, and then add labels and instructions. Repeat as needed.
- 3. On the **Package** menu, click **Save**.
- 4. In the **Package name** box, type a name for the template and optionally any passwords.
- 5. In the **Location** list, click **Structured Storage File**.
- 6. In the **File name** box, type a name for the package template and change the file suffix from .dts to .dtt.
- 7. Click the browse (...) button to select a directory location to save the template.

# **DTS Package Elements**

The following are procedures for adding, configuring, and using Data Transformation Services (DTS) tasks, transformations, connections, and workflow items.

# How to create a Transform Data task (Enterprise Manager)

## To create a Transform Data Task

1. On the Data Transformation Services (DTS) Designer design sheet, click the connection you want to use as a source, and then CTRL-click the connection you want to use as a destination.

You must have your source and destination connections defined before configuring a Transform Data task.

2. From the **Task** toolbar, drag a Transform Data task to the design sheet.

## See Also

**DTS** Connections

Tasks That Transform Data

Transform Data Task

# How to configure the connections for a Transform Data task (Enterprise Manager)

## To configure the connections for a Transform Data task

- 1. Point to the Transform Data task on the Data Transformation Services (DTS) Designer design sheet until the cursor changes, and then double-click to open the **Transform Data Task Properties** dialog box.
- 2. On the **Source** tab, type a description for the task.
- 3. Under **Connection**, do one of the following:
  - Click **Table** / **View** and select a table or view from the list.
  - Click SQL query. You can enter the query text in the box or click Build Query to create the query with DTS Query Designer. If you create a query with input parameters, click Parameters to assign the parameters to DTS package global variables.
- 4. Click the **Destination** tab, and then do one of the following:
  - In the **Table name** list, select a destination table.
  - Create a new table by clicking **Create** and editing the CREATE TABLE script in the **Create Destination Table** dialog box.

## See Also

DTS Connections

**DTS Transformations** 

Tasks That Transform Data

Transform Data Task

## How to configure a new transformation for a Transform Data task (Enterprise Manager)

#### To configure a new transformation for a Transform Data Task

- 1. After configuring your source and destination connections, click the **Transformations** tab.
- 2. Do one of the following:
  - If the columns you want are not already mapped to another transformation, click **New**.
  - If the columns you want are already mapped, click on the mapping line for the transformation, click **Delete**, and then click **New**.
- 3. In the **Create New Transformation** dialog box, click the type of transformation you want to add. The **Transformation Options** dialog box is displayed.
- 4. Click the **Source Columns** tab, and then use the arrow buttons to select columns for the transformation.
- 5. Click the **Destination Columns** tab, and then use the arrow buttons to select columns for the transformation.
- 6. Click the **General** tab, click **Properties**, and then accept or edit the properties for the transformation you selected.
- 7. Repeat steps 2 through 6 for each transformation you want to create.

See Also

Mapping Column Transformations DTS Transformations Tasks That Transform Data Transform Data Task

# How to activate the multiphase data pump feature (Enterprise Manager)

#### To activate the multiphase data pump feature

- 1. In SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **Properties**.
- 2. Under **Designer**, select the **Show multi-phase pump in DTS Designer** check box.

### See Also

Multiphase Data Pump Functionality

### How to add a multiphase data pump transformation function using an ActiveX script (Enterprise Manager)

To add a multiphase data pump transformation function using an ActiveX script

- 1. After activating the multiphase data pump feature and configuring your source and destination connections, click the **Transformations** tab.
- 2. In the **Phases filter** list, click the data pump phase you want to add.

The **Phases filter** list displays all the transformations configured for a specific data pump phase. By default, the **Row transform** phase is selected.

- 3. Do one of the following:
  - If the columns you want are not already mapped to another transformation, click **New**.
  - If the columns you want are already mapped, click on the mapping line for the transformation, click **Delete**, and then click **New**.
- 4. In the **Create New Transformation** dialog box, click **ActiveX Script**.
- 5. Click the **Source Columns** tab, and then use the arrow buttons to select columns for the transformation.
- 6. Click the **Destination Columns** tab, and then use the arrow buttons to include a column in the transformation.
- 7. Click the **General** tab, click **Properties**, and then click the **Phases** tab.

- 8. Select the data pump phases for which you will be adding Microsoft® ActiveX® script functions, and then for the selected phases, type the names of those functions in their respective boxes.
- 9. In the **ActiveX Script** text box, enter the ActiveX script code for each function.

The function names specified in the **ActiveX Script text box** must match those entered on the **Phases** tab.

### See Also

ActiveX Script Transformation

Multiphase Data Pump Functionality

Tasks That Transform Data

## How to call a COM object that customizes one or more data pump phases (Enterprise Manager)

#### To call a COM object that customizes one or more data pump phases

- 1. After activating the multiphase data pump feature and after configuring your source and destination connections, click the **Transformations** tab.
- 2. In the **Phases filter** list, click the data pump phase you want to customize.

The **Phases filter** list displays all the transformations configured for a specific data pump phase. By default, the **Row transform** phase is selected.

- 3. Do one of the following:
  - If the columns you want are not already mapped to another transformation, click **New**.
  - If the columns you want are already mapped, click on the mapping line for the transformation and click **Delete**, and then click **New**.
- 4. In the **Create New Transformation** dialog box, click the entry corresponding to the custom transformation (COM object) that will be called.

**Note** If the COM object has been installed on your computer but does not appear in the **Create New Transform** dialog box, and you have enabled Data Transformation Services (DTS) caching, you must refresh the cache. In SQL Server Enterprise Manager, right-click **Data Transformation Services**, and then click **Properties**. Under **Cache**, click **Refresh Cache**.

5. Click the **Source Columns** tab, and then use the arrow buttons to

select columns for the transformation.

- 6. Click the **Destination Columns** tab, and then use the arrow buttons to include a column in the transformation.
- 7. Click the **Phases** tab, and then click the data pump phases that will call the custom transformation you are supplying.

**Note** If the custom transformation includes a user interface, you can click the **General** tab and then click **Properties** to enter any additional information for the transformation.

### See Also

Multiphase Data Pump Functionality

Tasks That Transform Data

# How to enable the Transform Data task fast load options (Enterprise Manager)

#### To enable the Transform Data Task fast load options

- 1. After configuring the connections and transformations for a Transform Data task, click the **Options** tab.
- 2. Select the **Use fast load** check box, and then select any of the other Microsoft® SQL Server<sup>™</sup> fast load options that you want to use.

The fast load options are only in effect when you use the Microsoft OLE DB Provider for SQL Server as the destination connection.

### See Also

DTS Transformations Tasks That Transform Data Transform Data Task

# How to configure the fast load batch options (Enterprise Manager)

#### To configure the fast load batch options

- 1. After configuring the connections and transformations for a Transform Data task, click the **Options** tab.
- 2. Select the **Use fast load** check box.

The fast load options are only in effect when you use the Microsoft® OLE DB Provider for SQL Server as the destination connection.

- 3. In **Insert batch size** box, enter a value for the batch size.
- 4. Optionally, select the **Commit final batch** check box if you want to commit all rows in the last batch that will be copied prior to an error.
- 5. In the **Max error count** box, enter a value to specify the number of row-level errors detected by the Transform Data task plus the number of batch failures that must be exceeded before data pump operation for the task is terminated.

#### See Also

DTS Transformations Tasks That Transform Data Transform Data Task

# How to configure the data pump exception files (Enterprise Manager)

#### To configure the data pump exception files

1. On the Data Transformation Services (DTS) design sheet, double-click a Transform Data task or Data Driven Query task.

You must have your source and destination connections defined before configuring a Transform Data task.

- 2. Click the **Options** tab.
- 3. Under **Exception file**, in the **Name** box, type a file path for the text file you want to use as an exception file, or click the browse (...) button to locate the file.

If you enter a file that does not exist, the file will be created when the step associated with this transformation task is run.

- 4. Optionally, if you want to use the Microsoft® SQL Server<sup>™</sup> 2000 data pump exception file options, then under **File type**, clear the **7.0 format** check box and select the exception files you want to generate.
- 5. Under **File format**, select any additional options for the exception file.

#### See Also

<u>DTS Connections</u> <u>Tasks That Transform Data</u> <u>Transform Data Task</u>

### How to add a DTS task to a DTS package (Enterprise Manager)

#### To add a DTS task to a DTS package in DTS Designer

• From the **Task** toolbar, drag the task onto the Data Transformation Services (DTS) design sheet.

### See Also

DTS Tasks

## How to add the Bulk Insert task to a DTS package (Enterprise Manager)

#### To add a Bulk Insert task to a DTS package in DTS Designer

1. From the **Connection** toolbar, drag a Microsoft® OLE DB Provider for SQL Server connection onto the Data Transformation Services (DTS) design sheet and configure the connection.

**Note** The Bulk Insert Task requires one Microsoft OLE DB Provider for SQL Server connection. If such a connection is already configured in the package, you can skip Step 1.

2. From the **Task** toolbar, drag a Bulk Insert Task to the design sheet.

#### See Also

**Bulk Insert Task** 

## How to add the Execute SQL task to a DTS package (Enterprise Manager)

#### To add the Execute SQL task to a DTS package in DTS Designer

1. From the **Connection** toolbar, drag a connection onto the Data Transformation Services (DTS) design sheet and configure the connection.

**Note** The Execute SQL task requires one connection. If a connection is already configured in the package and you can access the data source through that connection, you can skip Step 1.

2. From the **Task** toolbar, drag an Execute SQL task to the design sheet.

#### See Also

Execute SQL Task

## How to execute a stored procedure with an input parameter (Enterprise Manager)

To execute a stored procedure with an input parameter using the Execute SQL task

- 1. In the **Execute SQL Task Properties** dialog box, in the **SQL statement** box, type the parameterized SQL stored procedure statement. For example: exec byRoyalty ?
- 2. Click **Parameters**, click the **Input Parameters** tab, and then assign a global variable and its value to the parameter.

## How to save row values into global variables (Enterprise Manager)

#### To save row values into global variables

- 1. From the **Connection** toolbar, drag a Microsoft® OLE DB Provider for SQL Server connection to the Data Transformation Services (DTS) design sheet.
- 2. In the **Connection Properties** dialog box, in the **Database** list, click **pubs**.
- 3. From the **Task** toolbar, drag an Execute SQL task to the design sheet.
- 4. In the **Execute SQL Task Properties** dialog box, in the **Existing connection** list, click the **pubs** connection just created.
- 5. In the SQL statement box, type the SQL code. For example: SELECT \* FROM titleauthor WHERE (royaltyper = '40')
- 6. Click **Parameters**, click **Create Global Variables**, and then enter the global variable names. For example: *o\_au\_id*, *o\_title\_id*, *o\_au\_order*, and *o\_royaltyper*.
- 7. Click the **Output Parameters** tab, click **Row Value**, and in the **Output Global Variables** column, click a row and select the global variable from the list to hold the column's data.

You can skip a column when saving values to a global variable. For example, if you do not want to store the value of the **title\_id** column, modify the **Output Global Variable** column to assign the **title\_id** 

column to **<none>**.

**Note** If the package has been executed and a value previously returned into the *o\_title\_id* global variable, setting the **title\_id** column to **<none>** will not reset or null the value of *o\_title\_id*. The global variable will contain the last value to which it was set. For more information, see <u>Using Global Variables with DTS Packages</u>.

### How to retrieve the row value data (Enterprise Manager)

#### To retrieve the row value data

- 1. From the **Task** toolbar, drag a Microsoft® ActiveX® Script task onto the Data Transformation Services (DTS) design sheet.
- 2. In the ActiveX Script Properties dialog box, after the Function Main() statement, type the following Microsoft Visual Basic® Scripting Editing (VBScript) code: MsgBox "The author ID is " & DTSGlobalVariables("o\_au\_id MsgBox "The title ID is " & DTSGlobalVariables("o\_title\_id" MsgBox "The au\_ord is " & DTSGlobalVariables("o\_au\_ord" MsgBox "The royalty is " & DTSGlobalVariables("o\_royaltyp

Main = DTSTaskExecResult\_Success

- 3. On the design sheet, click the Execute SQL task, and then CTRL-click the ActiveX Script task.
- 4. On the **Workflow** menu, click **On Success** or **On Completion**.

## How to save an entire rowset into a global variable (Enterprise Manager)

To save an entire result set of a SELECT statement into a global variable using the Execute SQL task

- 1. From the **Connection** toolbar, drag a Microsoft® OLE DB Provider for SQL Server connection to the design sheet.
- 2. In the **Database** list, click **pubs**.
- 3. From the **Task** toolbar, drag an Execute SQL task to the Data Transformation Services (DTS) design sheet.
- 4. In the **Execute SQL Properties** dialog box, in the **Existing connection** list, click the **pubs** connection just created.
- 5. In the SQL statement box, type the following: SELECT \* FROM titleauthor
- 6. Click **Parameters**, and then click **Create Global Variables**.
- In the Name list, type Authors, and then in the Type list, click <other>. Leave the Value box empty.
- 8. Click the **Output Parameters** tab, click **Rowset**, and then in the **Output Parameter Type** list, select the **Authors** global variable.

This procedure assigns all records returned from the SELECT statement to be stored in the **Authors** global variable.

## How to retrieve rowset data stored in a global variable (Enterprise Manager)

### To retrieve the rowset data stored in a global variable using the Execute SQL task

- 1. From the **Task** toolbar, drag a Microsoft® ActiveX® Script task onto the Data Transformation Services (DTS) design sheet.
- 2. In the ActiveX Script Properties dialog box, after the Function Main() statement, type the following Microsoft Visual Basic® Scripting Edition (VBScript) code: dim countr dim RS set RS = CreateObject("ADODB.Recordset") set RS = DTSGlobalVariables("Authors").value

```
for countr = 1 to RS.RecordCount
    MsgBox "The author ID is " & RS.Fields("au_id").value
    RS.MoveNext
    Next
```

Main = DTSTaskExecResult\_Success

- 3. On the design sheet, click the Execute SQL task, and then CTRL-click the ActiveX Script task.
- 4. On the Workflow menu, click **On Success** or **On Completion**.

The **au\_id** column for each row returned from the SELECT statement is displayed.

## How to send a message with the Message Queue task (Enterprise Manager)

#### To send a message with the Message Queue task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type a label that identifies the task on the design sheet.
- 3. In the **Message** list, click **Receive messages**.
- 4. In the Queue box, type the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the queue from which you will be reading messages.
- 5. Click **Add**, and then in the **Message type** list, select a type of message queue and configure each message:
  - Select **String Message**, and in the **String Message** box, type the message.
  - Select **Data File Message**, and in the **File Name** box, type the path of the file to send the message to, or click the browse (...) button to locate the file.
  - Select **Global Variables Message**. To add an existing package global variable, click **New**, and then in the name list, click a global variable. To create a global variable that does not exist in the package for use as a message, click **Create Global Variables**.

## How to receive a string message with the Message Queue task (Enterprise Manager)

#### To receive a string message with the Message Queue task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, enter a label that identifies the task on the design sheet.
- 3. In the **Message** list, click **Receive messages**.
- In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the queue from which you will be reading messages.
- 5. In the **Message type** list, click **String message**, and then under **Compare**, click an option for filtering the message.
- 6. If you click **Exact Match**, **Ignore Case**, or **Containing** in the **Compare String** box, type the search text.
- 7. Select the **Remove from message queue** check box to delete any received message from the queue. Clear this check box to leave the message on the queue after task completion.
- 8. Clear the **Timeout after** check box to wait indefinitely for an acceptable message or select the **Timeout after** check box to enter a timeout interval.

The timeout interval may take any value from 1 through 9999 (in seconds).

## How to receive a Data File Message with the Message Queue task (Enterprise Manager)

#### To receive a data file message with the Message Queue task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, enter a label that identifies the task on the design sheet.
- 3. In the **Message** list, click **Receive messages**.
- 4. In the **Queue** box, enter the name of the *computer\_name\queue\_type\$\queue\_name* combination that identifies the queue from which you will be reading messages.
- 5. In the **Message type** list, click **Data File Message**.
- 6. In the **Save file as** box, enter the path of a file or directory on your computer, or click the browse (...) button to locate the file.
- 7. Under **Only receive message from a specific package or version**, click a filter option.
- 8. If you select **From package** or **From version**, set **Identifier** to the globally unique identifier (GUID) string that identifies the proper DTS package or version. Use the browse (...) button to search for available packages and versions.

9. Clear the **Timeout after** check box to wait indefinitely for an acceptable message or select the **Timeout after** check box to enter a timeout interval. The value can range from 1 through 9999 (in seconds).

### How to receive a global variables message with the Message Queue task (Enterprise Manager)

To receive a global variables message with the Message Queue task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, enter a label that identifies the task on the design sheet.
- 3. In the **Message** list, click **Receive messages**.
- In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the queue from which you will be reading messages.
- 5. In the **Message type** box, click **Global Variables Message**.
- 6. Under **Only receive message from a specific package or version,** click a filter option.
- 7. If you selected **From package** or **From version**, set **Identifier** to the globally unique identifier (GUID) string that identifies the proper DTS package or version. Use the browse (...) button to search for available packages and versions.
- 8. Clear the **Timeout after** check box to wait indefinitely for an acceptable message. Select the **Timeout after** check box to enter a timeout interval. The value can range from 1 through 9999 (in seconds).

### How to configure the Ask For Facts task (Enterprise Manager)

#### **To configure the Ask For Facts task**

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type **Ask For Facts**.
- 3. In the **Message** box, click **Send Message**.
- 4. In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the location to which these messages will be sent.
- 5. Under **Messages to be sent**, click **New**, and then in the **Message type** box, click **String Message**.
- 6. In the **String Message** box, type **Summarize shipments**.
- 7. On the design sheet, right-click **Ask For Facts**, point to **Workflow**, and then click **Workflow Properties**.
- 8. Click the **Options** tab, and then clear the following check boxes:
  - Join transaction if present
  - Commit transaction on successful completion of this step
  - Rollback transaction on failure

### How to configure the Wait For Trigger task (Enterprise Manager)

#### To configure the Wait For Trigger task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type **Wait for Trigger**.
- 3. In the **Message** list, click **Receive Message**. In the **Queue** box, enter the same *computer\_name\queue\_type\$\queue\_name* combination as that entered for the **Ask for Facts** task.
- 4. In the **Message type** list, click **String Message**.
- 5. Under Compare, click **Exact Match**, and then in the **Compare String** box, type **Summarize shipments**.
- 6. Select the **Remove from message queue** check box.
- 7. On the design sheet, right-click **Wait for Trigger**, point to **Workflow**, and then click **Workflow Properties**.
- 8. In the **Workflow Properties** dialog box, click the **Options** tab, and then clear the following check boxes:
  - Join transaction if present
  - Commit transaction on successful completion of this step

• Rollback transaction on failure

# How to create and configure Add New Employees (Enterprise Manager)

To create and configure the Add New Employees package

- 1. In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **New Package**.
- 2. On the **Package** menu, click **Save**.
- 3. In the **Package name** box, type **Add New Employees**, and in the **Location** list, click **SQL Server**.
- 4. Right-click the Data Transformation Services (DTS) design sheet, and then click **Package Properties**.
- 5. Click the **Advanced** tab, select the **Use transactions** check box, and then clear the **Commit on successful package completion** check box.
- 6. Click the **Logging** tab, and then under **Error** handling, clear the **Fail package on first error** check box.

### How to configure the New Employee task (Enterprise Manager)

#### To configure the New Employee task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type **New Employee**.
- 3. In the **Message** box, click **Receive Message**.
- 4. In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination to identify the queue at which global variable messages will be received.
- 5. In the **Message type** list, click **Global Variables Message**, and then click **No filter**.
- 6. Specify the following options:
  - Select the **Remove from message queue** check box or the DTS package will loop indefinitely as it repeatedly attempts to add the same employee.
  - Select the **Timeout after** check box to force package termination after the queue is emptied, and then in the **seconds** box, type **1**.
- 7. On the design sheet, right-click **New Employee**, point to **Workflow**, and then select **Workflow Properties**.
- 8. In the **Workflow Properties** dialog box, click the **Options** tab, and

then clear the following check boxes:

- Join transaction if present
- Commit transaction on successful completion of this step
- Rollback transaction on failure

### How to configure the Transform Data task for Global Variable Messages (Enterprise Manager)

To configure the Transform Data task for global variable messages

- 1. Click **Not Used**, and then CTRL-click **Corporate**.
- 2. On the **Task** toolbar, click **Transform Data Task**, and then on the Data Transformation Services (DTS) design sheet, double-click the resulting arrow.
- 3. In the **Data Transformation Properties** dialog box, do the following:
  - a. In the **Description** box, type **Insert One Employee**.
  - b. Under **Connection**, click **SQL query**, and then in the text box, type **SELECT** 'xxx' AS xxx.

This generates a source rowset with one row in it. As a result, the insert will be attempted exactly once.

- 4. Click the **Destination** tab, and then in the **Table name** list, select the **Employee** table.
- 5. Click the **Transformations** tab, and then do the following:
  - a. Click **Delete All** to clear any default transformations.
  - b. In the **Destination** list, click **EmployeeID**, and then CTRLclick **EmployeeName**.
  - c. Click New.
- 6. In the **Create New Transformation** dialog box, click **ActiveX Script**.

- 7. In the **Transformation Options** dialog box, click **Properties**.
- 8. In the Active X Script Transformation Properties dialog box, under Entry function, type the following: DTSDestination ("EmployeeID") = DTSGlobalVariables("ID" DTSDestination("EmployeeName") = DTSGlobalVariables("I
- 9. Right-click the Transform Data task, and then click **Workflow Properties**.
- 10. In the **Workflow Properties** dialog box, click the **Options** tab, and then do the following:
  - Select the **Join transaction if present** check box.
  - Select the **Commit transaction on successful completion of this step** check box.
  - Clear the **Rollback transaction on failure** check box.
- 11. Click the **Precedence** tab, and then add a new entry with **Source Step** set to **New Employee** and **Precedence** set to **Success**.

### How to configure the Log Bad Update task (Enterprise Manager)

#### To configure the Log Bad Update task

- 1. From the Task toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type Log Bad Update.
- 3. In the **Message** list, click **Send Messages**.
- 4. In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the location to which bad update messages will be sent.
- 5. Click **Add**, and then in the **Message type** list, click **Global Variables Message**.
- 6. Add the global variables **ID** and **Name** by doing the following:
  - a. Click **Create Global Variables**.
  - b. In the **Global Variables** dialog box, type **ID** under a blank entry in the **Variables** list, then click **New** to create a new entry and enter **Name**.
  - c. In the **Message Queue Message Properties** dialog box, click **New**, click in the box under **Name**, and then select **ID** from the list. Repeat this step for the global variable **Name**.

Note Entries for Type and Value have no effect in this

application. You do not need to enter values.

- 7. On the design sheet, right-click **Log Bad Update**, point to **Workflow**, and then click **Workflow Properties**.
- 8. Click the **Options** tab, and then do the following:
  - Select the **Join transaction if present** check box.
  - Select the **Commit transaction on successful completion of this step** check box.
  - Clear the **Rollback transaction on failure** check box.
- 9. Click the **Precedence** tab, and then add a new entry with **Source Step** set to **Insert One Employee** and **Precedence** set to **Failure**.

### How to configure one Loop task (Enterprise Manager)

#### To configure one Loop task

- 1. On the Data Transformation Services (DTS) design sheet, right-click **New Employee**, point to **Workflow**, and then click **Workflow Properties**.
- 2. In the **Workflow Properties** dialog box, click the **Options** tab and note the name of the step in the **Name** box.

It is likely to be **DTSStep\_DTSMessageQueueTask\_1**. You will need to click this name in another box later in this procedure.

- 3. From the **Task** toolbar, drag a Dynamic Properties task onto the design sheet.
- 4. In the **Dynamic Properties Task Properties** dialog box, in the **Description** box, type **Loop**, and then click **Add**.
- In the tree display in the left pane, expand Steps, and then expand DTSStep\_DTSMessageQueueTask\_1 (the name you noted in Step 2).
- 6. In the right pane, under **Property name**, double-click **ExecutionStatus**.
- 7. In the **Add/Edit Assignment** dialog box, in the **Source** list, click **Constant**, and then in the **Constant** box, type **1**.

1 is the value assigned to DTSStepExecStat\_Waiting.

8. On the design sheet, right-click **Loop**, point to **Workflow**, and then click **Workflow Properties**.

9. Click the **Precedence** tab, and then add a new entry with **Source Step** set to either **Insert One Employee** or **Log Bad Update** and **Precedence** set to **Success**.

# How to create and configure the Load Expenses package (Enterprise Manager)

#### To create and configure the Load Expenses package

- 1. In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **New Package**.
- 2. On the **Package** menu, click **Save**.
- 3. In the **Package Name** box, type **Load Expenses**.
- 4. Right-click the Data Transformation Services (DTS) design sheet, and then click **Package Properties**.
- 5. Click the **Logging** tab, and then clear the **Fail package on first error** check box.
- 6. Click the **Advanced** tab, select the **Use transactions** check box, and then clear the **Commit on successful package completion** check box.

# How to configure the Spreadsheet Wait task (Enterprise Manager)

#### To configure the Spreadsheet Wait task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message** list, click **Receive Message**.
- 3. In the **Queue** box, enter the name of the *computer\_name\queue\_type\$\queue\_name* combination that identifies the queue where expense worksheets will be sent.
- 4. In the **Message Type** list, click **Data File Message**.
- 5. In the **Save file as** box, type C:\Temp\Expense.xls.
- 6. Do the following:
  - Select the **Overwrite** check box to prevent the first worksheet from blocking further uploads.
  - Select the **Remove from message queue** check box or the package will loop indefinitely as it repeatedly attempts to load the same worksheet.
  - Clear the **Timeout after** check box because this package runs until canceled.
- 7. On the design sheet, right-click **Wait**, point to **Workflow**, and then click **Workflow Properties**.

- 8. Click the **Options** tab, and then clear the following check boxes:
  - Join transaction if present
  - Commit transaction on successful completion of this step
  - Rollback transaction on failure

# How to configure the Delete Raw Data task (Enterprise Manager)

#### To configure the Delete Raw Data task

- 1. From the **Task** toolbar, drag an Execute SQL task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Execute SQL Task Properties** dialog box, in the **Description** box, type **Delete Raw Data**.
- 3. In the **Existing Connection** list, click **Raw Data**.
- 4. In the **SQL Statement** text box, type the following: DELETE FROM RawExpense.
- 5. On the design sheet, right-click **Delete Raw Data**, point to **Workflow**, and then click **Workflow Properties**.
- 6. Click the **Options** tab, and then clear the following check boxes:
  - Join transaction if present
  - Commit transaction on successful completion of this step
  - Rollback transaction on failure
- 7. Click the **Precedence** tab, and then add a new entry with **Source Step** set to **Spreadsheet Wait** and set **Precedence** to **Success**.

# How to configure the Load Raw Data task (Enterprise Manager)

#### **To configure the Load Raw Data task**

- 1. Click **Expense Report**, and then CTRL-click **Raw Data**.
- 2. On the **Task** toolbar, click **Transform Data Task**, and then on the Data Transformation Services (DTS) design sheet, double-click the resulting arrow.
- 3. In the **Description** box, type **Load Raw Data**, and in the **Table/View** list, click **Expenses** to load data from the proper spreadsheet range.
- 4. Click the **Destination** tab, and then in the **Table name** list, click **RawExpense**.
- 5. Click the **Transformations** tab, and then click **OK**.

Clicking **OK** without editing the transformations saves the default column mappings.

- 6. On the design sheet, right-click the Transform Data task, point to **Workflow**, and then click **Workflow Properties**.
- 7. Click the **Options** tab, and then do the following:
  - Clear the **Commit transaction on successful completion of this step** check box.
  - Clear the **Rollback transaction on failure** check box.
  - Select the **Join transaction if present** check box.
- 8. Click the **Precedence** tab, and then add a new entry with **Source Step**

set to **Delete Raw Data** and **Precedence** set to **Success**.

# How to configure the Load Filtered Data task (Enterprise Manager)

#### To configure the Load Filtered Data task

- 1. Click **Raw Data**, and then CTRL-click **Corporate**.
- 2. On the **Task** toolbar, click **Transform Data Task**, and then on the Data Transformation Services (DTS) design sheet, double-click the resulting arrow.
- 3. In the **Description** box, type **Load Final Data**.
- Under Connection, click SQL query, and then in the text box, type the following SQL statement: SELECT FROM RawExpense WHERE ExpenseDate IS NOT

This generates a source rowset without any null rows.

- 5. Click the **Destination** tab, and then in the **Table name** list, click **Expense**.
- 6. Click the **Transformations** tab, and then click **OK**.

Clicking **OK** without editing the transformations saves the default column mappings.

- 7. On the design sheet, right-click the Transform Data task, point to **Workflow**, and then click **Workflow Properties**.
- 8. Click the **Options** tab, and then do the following:
  - Select the **Commit transaction on successful completion of this step** check box.

- Select the **Join transaction if present** check box.
- Clear the **Rollback transaction on failure** check box.
- 9. Click the **Precedence** tab, and then add a new entry with **Source Step** set to **Load Raw Data** and **Precedence** set to **Success**.

# How to configure the Failed Expense Load (or Failed XLS Load) task (Enterprise Manager)

To configure the Failed Expense Load (or Failed XLS Load) task

- 1. From the **Task** toolbar, drag a Message Queue task onto the Data Transformation Services (DTS) design sheet.
- 2. In the **Message Queue Task Properties** dialog box, in the **Description** box, type **Failed Expense Load** (or **Failed XLS Load**).
- 3. In the **Message** list, click **Send Message**.
- 4. In the Queue box, enter the name of the computer\_name\queue\_type\$\queue\_name combination that identifies the location to which unloadable worksheets will be sent.
- 5. In the **Messages to be sent** list, click **Add**.
- 6. In the **Message type** box, click **Data File Message**, and in the **File Name** box, type C:\Temp\Expense.xls.
- 7. On the design sheet, right-click **Failed Expense Load** (or **Failed XLS Load**), point to **Workflow**, and then click **Workflow Properties**.
- 8. Click the **Options** tab, and then select the following check boxes:
  - Commit transaction on successful completion of this step
  - Join transaction if present

- Rollback transaction on failure
- 9. Click the **Precedence** tab, and then add a new entry with **Source Step** set to **Load Filtered Data** (or **Load Raw Data**) and **Precedence** set to **Failure**.

# How to create and configure three Loop tasks (Enterprise Manager)

#### To create and configure three Loop tasks

- 1. On the Data Transformation Services (DTS) design sheet, right-click **Spreadsheet Wait**, point to **Workflow**, and then click **Workflow Properties**.
- 2. In the **Workflow Properties** dialog box, click the **Options** tab and note the name of the step in the **Name** box.

It is likely to be **DTSStep\_DTSMessageQueueTask\_1**. You will need to click this name in another box later in this procedure.

- 3. From the **Task** toolbar, drag a Dynamic Properties task onto the design sheet.
- 4. In the **Description** box, type **Loop**, and then click **Add**.
- In the tree display in the left pane, expand Steps, and then expand DTSStep\_DTSMessageQueueTask\_1 (the name you noted in Step 2).
- 6. In the right pane, under **Property** name, double-click **Execution Status.**
- 7. In the **Add/Edit Assignment** dialog box, in the **Source** list, click **Constant**, and then in the **Constant** box, type **1**.

1 is the value assigned to **DTSStepExecStat\_Waiting**.

- 8. On the design sheet, do the following:
  - Right-click **Loop**, and then click **Copy**.

- Right-click the design sheet, and then click **Paste**. Repeat this paste once.
- Position the three **Loop** tasks in a column on the right side of the design sheet.
- 9. Right-click each of these pasted **Loop** tasks, point to **Workflow**, and then click **Workflow Properties**.
- Click the Precedence tab, and then add a new entry with Source Step set to the proper step: Failed XLS Load, Failed Expense Load, or Load Filtered Data. Set Precedence to Success.

# How to convert the format of a Date Time String transformation (Enterprise Manager)

To convert the format of a Date Time String transformation

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the date or time to be modified, and then click the **Destination** column where you want the modified string to be placed.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**.
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, click **DateTime String**.
- 4. Click the **General** tab, and then click **Properties**.
  - 5. In the **Date Format** list, select the format you want.
- 6. Click **Naming** to display the **Calendar Names** dialog box, where you can select long or short day or month names and the A.M. and P.M. designators you want.
- 7. In the **Language** list, select the language you want, and then click **Set Language Defaults**.

### See Also

Date Time String Transformation

# How to convert a string to lowercase characters (Enterprise Manager)

To convert a string to lowercase characters

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the string to be modified, and then click the **Destination** column where you want the modified string to be placed.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**.
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, select **Lowercase String**.

### See Also

Lowercase String Transformation

# How to convert a string to uppercase characters (Enterprise Manager)

To convert a string to uppercase characters

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the string to be modified, and then click the **Destination** column where you want the modified string to be placed.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, select **Uppercase String**.

### See Also

**Uppercase String Transformation** 

# How to perform a Middle of String transformation (Enterprise Manager)

### To perform a Middle of String transformation

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the string to be modified, and then click the **Destination** column where you want the modified string to be placed.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**.
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, select **Middle of String**.
- 4. Click the **General** tab, and then click **Properties**.
- 5. In the **Start position (1 based)** box, type or select the first character position occupied by the substring.
- 6. Optionally, if you want to remove characters from the end of the source string, select the **Limit number of characters to** check box and enter a maximum substring length.

### See Also

<u>Middle of String Transformation</u> Transform Data Task

# How to perform a Trim String transformation (Enterprise Manager)

### To perform a Trim String transformation

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the string to be modified, and then click the **Destination** column where you want the modified string to be placed.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**.
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, select **Trim String**.
  - 4. Click the **General** tab, and then click **Properties**.

### See Also

Trim String Transformation

# How to perform a Read File transformation (Enterprise Manager)

### To perform a Read File transformation

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the file specification information, and then click the **Destination** column to which you want the contents of the file copied.
- 2. Do one of the following:
  - If there is a mapping arrow connecting the two columns, click **Delete**, and then click **New**.
  - If there is no mapping arrow, click **New**.
- 3. In the **Create New Transformation** dialog box, click **Read File**.
- 4. Click the **General** tab, and then click **Properties**.
- 5. In the **Read File Transformation** dialog box, do the following:
  - In the **Directory** box, type the name of the directory from which the files are to be read.
  - In the **File type** list, click the file type you want.
  - Select the **Error if file not found** check box to fail the step when no file matches the source column. Clear this check box to null the destination column.

Read File Transformation

# How to perform a Write File transformation (Enterprise Manager)

#### To perform a Write File transformation

- 1. On the **Transformations** tab of the **Transform Data Task Properties** or **Data Driven Query Task Properties** dialog box, click the **Source** column containing the file name column and the **Source** column containing the data column.
- 2. Click **New**, and in the **Create New Transformation** dialog box, click **Write File**.
- 3. Click the **General** tab, and then click **Properties**.
- 4. In the **Write File Transformation Properties** dialog box, do the following:
  - In the **Directory** box, type the name of the directory in which the files are to be saved.
  - In the **File type** list, click the file type you want.
  - In the **File name column** list, click the column that contains the file names.
  - Under **Handle existing file**, click the option that you want.

#### See Also

<u>Write File Transformation</u> <u>Transform Data Task</u>

### How to create a connection (Enterprise Manager)

#### To create a connection in DTS Designer

- 1. On the **Connection** toolbar, drag the connection you want onto the Data Transformation Services (DTS) Designer design sheet.
- 2. Do one of the following:
  - Click **New connection**.
  - Click **Existing connection**, and then click an available connection from the list.
- 3. Complete the rest of the connection configuration information.

The types of information will vary according to the particular data provider you choose.

# How to create a data link with run-time resolution (Enterprise Manager)

### To create a data link with run-time resolution

- 1. In the **Connection Properties** dialog box, in the **Data Source** list, click **Microsoft Data Link**.
- 2. Select the **Always read properties from UDL file** check box.

If the check box is cleared, connection properties must be edited through the Data Transformation Services (DTS) package.

3. Click **Properties** to display the **Data Link Properties** dialog box.

### See Also

Data Link Connection

# **DTS Package Management**

The topics in this section contain procedures for managing Data Transformation Services (DTS) packages from SQL Server Enterprise Manager and from within DTS tools.

# How to create a DTS package using DTS Designer (Enterprise Manager)

To create a DTS package using DTS Designer

• In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **New Package**.

Note Your choice of Local Packages or Meta Data Services Packages determines the format in which the file is saved by default (Local packages are saved to Microsoft® SQL Server<sup>™</sup> and Meta Data Services packages are saved to SQL Server 2000 Meta Data Services).

### See Also

Managing a DTS Package

# How to create a DTS package using the DTS Import/Export Wizard (Enterprise Manager)

To create a new package using the DTS Import/Export Wizard

• In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, point to **All Tasks**, and then click **Import Data** or **Export Data**.

See Also

dtswiz Utility

DTS Import/Export Wizard

Managing a DTS Package

# How to edit a DTS package saved to SQL Server or Meta Data Services (Enterprise Manager)

To edit a DTS package saved to SQL Server or Meta Data Services

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services.**
- 2. Do one of the following:
  - Click **Local Packages** for Data Transformation Services (DTS) packages saved to Microsoft® SQL Server<sup>™</sup>.
  - Click **Meta Data Services Packages** for packages saved to SQL Server 2000 Meta Data Services.
- 3. In the details pane, double-click the DTS package you want to open in DTS Designer.

# See Also

Managing a DTS Package

Saving a DTS Package to Meta Data Services

Saving a DTS Package to SQL Server

# How to edit a DTS package saved to a structured storage file (Enterprise Manager)

To edit a DTS package saved to a structured storage file

- 1. In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **Open Package**.
- 2. In the **Select File** dialog box, browse for the file you want, click the file, and then click **Open**.
- 3. If the file contains multiple packages or multiple package versions, the **Select Package** dialog box appears. Double-click a Data Transformation Services (DTS) package or package version. If you want the most recent version of a package, click the package node or the latest version node.

### See Also

Saving a DTS Package to a Structured Storage File

Managing a DTS Package

## How to delete a DTS package (Enterprise Manager)

#### To delete a DTS package saved to SQL Server or Meta Data Services

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services.**
- 2. Do one of the following:
  - Click **Local Packages** for Data Transformation Services (DTS) packages saved to Microsoft® SQL Server<sup>™</sup>.
  - Click **Meta Data Services Packages** for packages saved to SQL Server 2000 Meta Data Services.
- 3. In the details pane, right-click a Data Transformation Services (DTS) package, and then click **Delete**.

This deletes all versions of the package.

#### To delete a DTS package version saved to SQL Server

- 1. In SQL Server Enterprise Manager console tree, expand **Local Packages**.
- 2. In the detail pane, right-click the package, and then click **Versions**.
- 3. In the **DTS Package Versions** dialog box, click the package version, and then click **Delete**.

#### To delete a DTS package saved to a file

- 1. Using a file manager such as Microsoft Windows Explorer, click the folder containing your DTS packages.
- 2. Right-click the desired package file, and then click **Delete**.

See Also

Managing a DTS Package

# How to execute a DTS package from SQL Server Enterprise Manager (Enterprise Manager)

To execute a DTS package from SQL Server Enterprise Manager

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services.**
- 2. Do one of the following:
  - Click **Local Packages** for Data Transformation Services (DTS) packages saved to Microsoft® SQL Server<sup>™</sup>.
  - Click **Meta Data Services Packages** for packages saved to SQL Server 2000 Meta Data Services.
- 3. In the details pane, right-click the package, and then click **Execute Package**.

### See Also

# How to execute a DTS package from DTS Designer (Enterprise Manager)

### To execute a DTS package from DTS Designer

• With the Data Transformation Services (DTS) package open in DTS Designer, click the **Execute** button on the toolbar.

# See Also

# How to execute a DTS package from the DTS Import/Export Wizard (Enterprise Manager)

To execute a DTS package from the DTS Import/Export Wizard

- 1. On the **Save, Schedule and Replicate Package** dialog box, click **Run immediately**, and then click **Next**.
- 2. On the **Completing the DTS Wizard** dialog box, click **Finish**.

### See Also

# How to execute a DTS package using the DTS Run utility (Command Prompt)

#### To execute a DTS package using the DTS Run utility

- 1. Open a command prompt window and type **dtsrunui** without any command switches.
- 2. In the **DTS Run** and **Advanced DTS Run** dialog boxes, enter any information for connection settings and logging.
- 3. Click **OK** when you are ready to execute the package.

#### See Also

**DTS Package Execution Utilities** 

# How to execute a DTS package using dtsrun (Command Prompt)

#### To execute a DTS package using dtsrun

• Open a command prompt window and type **dtsrun** with any necessary and optional command switches.

### See Also

dtsrun Utility

# How to save a DTS package to SQL Server (Enterprise Manager)

#### To save a DTS package to SQL Server

- 1. In Data Transformation Services (DTS) Designer, on the toolbar, click the **Save** button.
- 2. In the **Location** list, click **SQL Server**.
- 3. Complete the rest of the required fields.

**Note** If the DTS package has already been saved to Microsoft® SQL Server<sup>™</sup>, the **Save DTS Package** dialog box will not appear and a new version will be saved. If the package has been saved to a structured storage or a Microsoft Visual Basic® file, or to SQL Server 2000 Meta Data Services, click **Save As** to save to SQL Server.

### See Also

Saving a DTS Package to SQL Server

# How to open a DTS package saved to SQL Server (Enterprise Manager)

#### To open a DTS package saved to SQL Server

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services**, and then click **Local Packages**.
- 2. Do one of the following:
  - Double-click the Data Transformation Services (DTS) package you want to open in DTS Designer.
  - Right-click the DTS package you want to open, and then click **Versions**. In the **DTS Package Versions** dialog box, click the package version you want, and then click **Edit**.

#### See Also

Saving a DTS Package to SQL Server

# How to save a DTS package to Meta Data Services (Enterprise Manager)

#### To save a DTS package to Meta Data Services

- 1. In Data Transformation Services (DTS) Designer, on the **Package** menu, click **Save**.
- 2. In the **Location** list, click **Meta Data Services**.
- 3. Complete the rest of the required fields.

**Note** If the package has already been saved to Microsoft® SQL Server<sup>™</sup> 2000 Meta Data Services, the **Save DTS Package** dialog box will not appear, and a new version will be saved. If the package has been saved to a structured storage or a Microsoft Visual Basic® file, or to SQL Server, click **Save As** to save to Meta Data Services.

### See Also

Saving a DTS Package to Meta Data Services

# How to open a DTS package saved to Meta Data Services (Enterprise Manager)

#### To open a DTS package saved to Meta Data Services

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services**, and then double-click **Meta Data Services Packages**.
- 2. Do one of the following:
  - Double-click the Data Transformation Services (DTS) package you want to open in DTS Designer.
  - Right-click the DTS package you want to open, and then click **Versions**. In the **DTS Package Versions** dialog box, click the package version you want, and then click **Edit**.

#### See Also

Saving a DTS Package to Meta Data Services

# How to save a DTS package to a structured storage file (Enterprise Manager)

#### To save a DTS package to a structured storage file

- 1. In Data Transformation Services (DTS) Designer, on the toolbar, click **Save**.
- 2. In the **Location** list, click **Structured Storage File**.
- 3. Complete the rest of the required fields.

**Note** If the DTS package has already been saved to a structured storage file, the **Save DTS Package** dialog box will not appear, and a new version will be saved. If the package has been saved to Microsoft® SQL Server<sup>™</sup>, SQL Server 2000 Meta Data Services, or a Microsoft Visual Basic® file, click **Save As** to save to a structured storage file.

### See Also

Saving a DTS Package to a Structured Storage File

# How to open a DTS package saved to a structured storage file (Enterprise Manager)

To open a DTS package saved to a structured storage file

- 1. In the SQL Server Enterprise Manager console tree, right-click **Data Transformation Services**, and then click **Open Package**.
- 2. In the **Select File** dialog box, click the .dts file you want, and then click **Open**.
- 3. If multiple Data Transformation Services (DTS) packages or package versions were saved, the **Select Package** dialog box appears. Click the package or package version you want to open.

#### See Also

Saving a DTS Package to a Structured Storage File

# How to schedule a DTS package using the Schedule Package option (Enterprise Manager)

To schedule a DTS package using the Schedule Package option

- 1. In the SQL Server Enterprise Manager console tree, expand **Data Transformation Services**, and then click either **Local Packages** or **Meta Data Services Packages**.
- 2. In the details pane, right-click the Data Transformation Services (DTS) package you want to schedule, and then click **Schedule Package**.
- 3. In the **Edit Recurring Job Schedule** dialog box, complete the required information.

**Note** The scheduled package will be executed by SQL Server Agent using the permissions specified during server registration. If Windows Authentication was used, then SQL Server Agent will attempt to load the package using its own security, which may not be sufficient to load the package. For more information see <u>Handling Package Security in DTS</u>.

### See Also

Scheduling a DTS Package for Execution

# How to schedule a DTS package using SQL Server Agent (Enterprise Manager)

#### To schedule a DTS package using SQL Server Agent

- 1. In SQL Server Enterprise Manager console tree, expand **Management**, and then click **SQL Server Agent**.
- 2. In the details pane, right-click **Jobs**, and then click **New Job**.
- 3. On the **General** tab, complete the information to configure the new job.
- 4. Click the **Steps** tab, click **New**, and then do the following:
  - In the **Step name** box, type a name.
  - In the **Type** list, click **Operating System Command** (CmdExec).
  - In the **Command** text box, type the **dtsrun** command for the package.

**Note** Scheduled packages are run by SQL Server Agent and, as such, do not have the same shared drive letters or the same permissions as the package creator.

#### See Also

Scheduling a DTS Package for Execution

# How to view or modify DTS package properties (Enterprise Manager)

#### To view or modify DTS package properties

- With the Data Transformation Services (DTS) package open in DTS Designer, right-click the design sheet, and then click **Package Properties**.
- 2. Click the tab you want, and then view or modify the values.

#### See Also

Viewing and Configuring DTS Package Properties

# How to use Disconnected Edit to modify DTS package properties (Enterprise Manager)

#### To use Disconnected Edit to modify DTS package properties

- 1. With the Data Transformation Services (DTS) package open in DTS Designer, right-click the design sheet, and then click **Disconnected Edit**.
- 2. In the **Edit All Package Properties** dialog box, expand the property nodes in the left pane, and then double-click the property group you want to edit.
- 3. In the right pane, select the property you want and click **Edit**.
- 4. In the **Type** list, choose a property, and then in the **Value** box, type a new property value.

**Note** Not all property values can be modified though the **Disconnected Edit** feature.

#### See Also

Viewing and Configuring DTS Package Properties

# How to save a DTS package to a Visual Basic file (Enterprise Manager)

To save a DTS package to a Visual Basic file using the DTS Import/Export Wizard

• On the Save, Schedule and Replicate Package screen, select the Save DTS Package check box, and then click Visual Basic File.

#### To save a DTS package to a Visual Basic file using DTS Designer

- 1. On the **Package** menu, click **Save As**.
- 2. In the **Location** list, click **Visual Basic File**.
- 3. In the **File Name** box, type the name of the Microsoft® Visual Basic® file.

#### See Also

Saving a DTS Package to a Visual Basic File

### How To View Package Logs (Enterprise Manager)

#### To view package logs

- 1. In SQL Server Enterprise Manager, expand **Data Transformation Services**.
- 2. Do one of the following:
  - Right-click Local Packages (if the Data Transformation Services (DTS) package log was saved to Microsoft® SQL Server<sup>™</sup>) and then click Package Logs.
  - Right-click **Meta Data Services Packages** (if the package log was saved to SQL Server 2000 Meta Data Services), and then click **Package Logs**.
  - Click **Local Packages** or **Meta Data Services Packages**, and in the details pane, right-click a package and click **Package Logs**.

#### See Also

Using DTS Package Logs

## How to Enable Package Logging (Enterprise Manager)

#### To enable package logging

- 1. Open the Data Transformation Services (DTS) package for which you want to create a log.
- 2. On the **Package** menu, click **Properties** to display the **DTS Package Properties** dialog box.
- 3. Do one of the following:
  - Save package logs to Microsoft<sup>®</sup> SQL Server<sup>™</sup> by clicking the Logging tab, selecting the Log package execution to SQLServer check box, and then clicking an available server on which to save the package logs.
  - Save package logs to SQL Server 2000 Meta Data Services by clicking the Advanced tab, and then selecting the Show lineage variables as source columns and Write lineage to repository check boxes. On the Package menu, click Save As, and then in the Save DTS Package dialog box, in the Location list, select Meta Data Services.

#### See Also

How To View Package Logs Using DTS Package Logs

# How to set a DTS package password (Enterprise Manager)

#### To set a DTS package password in DTS Designer

- 1. On the **Package** menu, click **Save** or **Save As**.
- 2. In the **Location** list, click either **SQL Server** or **Structured Storage File**.
- 3. Do one of the following:
  - Enter an **Owner** password. Assigning an **Owner** password puts limits on who can both edit and run the package.
  - Enter a **User** password. Assigning a **User** password puts limits only on who can edit the package. If you create a **User** password, you must also create an **Owner** password.

## See Also

Handling Package Security in DTS

# How to modify the persisting of authentication information (Enterprise Manager)

To modify the persisting of authentication information in a DTS package

- 1. On the Data Transformation Services (DTS) Designer design sheet, double-click a connection.
- 2. In the **Connection Properties** dialog box, click **Advanced**.
- 3. Under the **Value** column, click the value for the **Persist Security Info** property.
- 4. Do one of the following:
  - Type **0** to disable the saving of authentication information with the DTS package.
  - Type **1** to persist the saving of authentication information.

### See Also

Handling Package Security in DTS

# How to select the Turn on just-in-time debugging option (Enterprise Manager)

#### To select the Turn on just-in-time debugging option

- 1. In the SQL Server Enterprise Manager console tree, right-click the **Data Transformation Services** node, and then click **Properties**.
- 2. Select the **Turn on just-in-time debugging** check box.

# How to add ActiveX workflow scripts in DTS Designer (Enterprise Manager)

#### To add ActiveX workflow scripts in DTS Designer

- 1. Right-click the task icon associated with step you want to configure, point to **Workflow**, and then click **Workflow Properties**.
- 2. Click the **Options** tab, select the **Use ActiveX Script** check box, and then click **Properties**.
- 3. In the Microsoft® ActiveX® Script text box, enter the scripting code for the workflow step.

# How to execute a single package step in DTS Designer (Enterprise Manager)

To execute a single package step in DTS Designer

- 1. On the **Task** toolbar, right-click the task you want to execute.
- 2. Click **Execute Step**.

# **Transact-SQL**

These procedures allow you to administer installations of SQL Server or administer SQL Server replication using Transact-SQL statements.

# **Administering SQL Server**

Microsoft® SQL Server<sup>™</sup> administration applications and their accompanying services are designed to assist the system administrator with all administrative tasks related to maintaining and monitoring server performance and activities.

# **Backing Up and Restoring Databases**

The backup and restore component of Microsoft® SQL Server<sup>™</sup> provides an important safeguard for protecting critical data stored in SQL Server databases. Understanding how to create and restore database, differential database, transaction log, and file and filegroup backups helps you implement this important safeguard.

# How to create a database backup (Transact-SQL)

#### To create a database backup

- 1. Execute the BACKUP DATABASE statement to create the database backup, specifying:
  - The name of the database to back up.
  - The backup device where the database backup will be written.

#### 2. Optionally, specify:

- The INIT clause to overwrite the backup media, and write the backup as the first file on the backup media. If no existing media header exists, one is automatically written.
- The SKIP and INIT clauses to overwrite the backup media even if there are either backups on the backup media that have not yet expired, or the media name does not match the name on the backup media.
- The FORMAT clause when using media for the first time to completely initialize the backup media and rewrite any existing media header.

The INIT clause is not required if the FORMAT clause is specified.

**IMPORTANT** Use extreme caution when using the FORMAT or INIT clauses of the BACKUP statement, as this will destroy any backups previously stored on the backup media.

### Examples

This example backs up the entire **MyNwind** database to tape:

```
USE MyNwind
GO
BACKUP DATABASE MyNwind
TO TAPE = '\\.\Tape0'
WITH FORMAT,
NAME = 'Full Backup of MyNwind'
GO
```

## See Also

sp\_addumpdevice

Database Backups

Appending Backup Sets

**Differential Database Backups** 

Backing Up the master Database

**Deleting a Database** 

Backing Up the model, msdb, and distribution Databases

**Overwriting Backup Media** 

**BACKUP** 

**Reducing Recovery Times** 

Transaction Log Backups

Initializing Backup Media

# How to restore a database backup (Transact-SQL)

#### To restore a database backup

**IMPORTANT** The system administrator restoring the database backup must be the only person currently using the database to be restored.

- 1. Execute the RESTORE DATABASE statement to restore the database backup, specifying:
  - The name of the database to restore.
  - The backup device from where the database backup will be restored.
  - The NORECOVERY clause if you have a transaction log or differential database backup to apply after restoring the database backup.
- 2. Optionally, specify:
  - The FILE clause to identify the backup set on the backup device to restore.

# Examples

This example restores the **MyNwind** database backup from tape:

```
USE master
GO
RESTORE DATABASE MyNwind
FROM TAPE = '\\.\Tape0'
GO
```

# See Also

Database Backups

Setting Database Options

**RESTORE** 

Identifying the Backup Set to Restore

Transaction Log Backups

**Differential Database Backups** 

Rebuilding the master Database

Restoring the model, msdb, and distribution Databases

Reducing Recovery Times

# How to restart an interrupted backup operation (Transact-SQL)

#### To restart an interrupted backup operation

- Execute the interrupted BACKUP statement again, specifying:
  - The same clauses used in the original BACKUP statement.
  - The RESTART clause.

# Examples

This example restarts an interrupted database backup operation:

-- Create a database backup of the MyNwind database BACKUP DATABASE MyNwind

TO MyNwind\_1

- -- The backup operation halts due to power outage.
- -- Repeat the original BACKUP statement specifying WITH RESTAR

BACKUP DATABASE MyNwind

TO MyNwind\_1 WITH RESTART

# See Also

BACKUP

Database Backups

## How to restart an interrupted restore operation (Transact-SQL)

#### To restart an interrupted restore operation

**IMPORTANT** The system administrator restoring the backup must be the only person currently using the database to be restored.

- Execute the interrupted RESTORE statement again, specifying:
  - The same clauses used in the original RESTORE statement.
  - The RESTART clause.

## Examples

This example restarts an interrupted restore operation:

```
-- Restore a database backup of the MyNwind database
RESTORE DATABASE MyNwind
FROM MyNwind_1
GO
-- The restore operation halted prematurely.
-- Repeat the original RESTORE statement specifying WITH RESTAF
RESTORE DATABASE MyNwind
FROM MyNwind_1
WITH RESTART
GO
```

### See Also

Database Backups

**RESTORE** 

**Copying Databases** 

# How to create a transaction log backup (Transact-SQL)

#### To create a transaction log backup

- 1. Execute the BACKUP LOG statement to back up the transaction log, specifying:
  - The name of the database to which the transaction log to back up belongs.
  - The backup device where the transaction log backup will be written.
- 2. Optionally, specify:
  - The INIT clause to overwrite the backup media, and write the backup as the first file on the backup media. If no existing media header exists, one is automatically written.
  - The SKIP and INIT clauses to overwrite the backup media even if there are either backups on the backup media that have not yet expired, or the media name does not match the name on the backup media.
  - The FORMAT clause, when using media for the first time, to completely initialize the backup media and rewrite any existing media header.

The INIT clause is not required if the FORMAT clause is specified.

**IMPORTANT** Use extreme caution when using the FORMAT or INIT clauses of the BACKUP statement as this will destroy any backups previously stored on the backup media.

### Examples

This example creates a transaction log backup for the **MyNwind** database to the previously created named backup device, **MyNwind\_log1**:

```
BACKUP LOG MyNwind
TO MyNwind_log1
GO
```

## See Also

BACKUP

Transaction Log Backups

Restoring a Database to a Prior State

**Reducing Recovery Times** 

File and Filegroup Backup and Restore

# How to back up the transaction log when the database is damaged (Transact-SQL)

#### To create a backup of the currently active transaction log

- 1. Execute the BACKUP LOG statement to back up the currently active transaction log, specifying:
  - The name of the database to which the transaction log to back up belongs.
  - The backup device where the transaction log backup will be written.
  - The NO\_TRUNCATE clause to back up the transaction log without truncating the inactive part of the transaction log.

This clause allows the active part of the transaction log to be backed up even if the database is inaccessible, provided that the transaction log file(s) is accessible and undamaged.

- 2. Optionally, specify:
  - The INIT clause to overwrite the backup media, and write the backup as the first file on the backup media. If no existing media header exists, one is automatically written.
  - The SKIP and INIT clauses to overwrite the backup media, even if there are either backups on the backup media that have not yet expired, or the media name does not match the name on the backup media.
  - The FORMAT clause, when using media for the first time, to completely initialize the backup media and rewrite any existing media header.

The INIT clause is not required if the FORMAT clause is specified.

**IMPORTANT** Use extreme caution when using the FORMAT or INIT clauses of the BACKUP statement as this will destroy any backups previously stored on the backup media.

# Examples

This example backs up the currently active transaction log for the **MyNwind** database even though **MyNwind** has been damaged and is inaccessible. The transaction log, however, is undamaged and accessible:

```
BACKUP LOG MyNwind
TO MyNwind_log1
WITH NO_TRUNCATE
GO
```

See Also

BACKUP

**Transaction Log Backups** 

Restoring a Database to a Prior State

**Reducing Recovery Times** 

File and Filegroup Backup and Restore

# How to apply a transaction log backup (Transact-SQL)

It is not possible to apply a transaction log backup:

- Unless the database or differential database backup preceding the transaction log backup is restored first.
- Unless all preceding transaction logs created since the database or differential database were backed up are applied first.
- If the database has already recovered and all outstanding transactions have either been rolled back or rolled forward.

#### To apply a transaction log backup

- 1. Execute the RESTORE LOG statement to apply the transaction log backup, specifying:
  - The name of the database to which the transaction log will be applied.
  - The backup device where the transaction log backup will be restored from.
  - The NORECOVERY clause if you have another transaction log backup to apply after the current one, otherwise specify the RECOVERY clause.
- 2. Repeat Step 1 for each transaction log backup you need to apply.

# Examples

# A. Applying a single transaction log backup

This example applies a transaction log backup to the **MyNwind** database.

```
RESTORE LOG MyNwind
FROM MyNwind_log1
WITH RECOVERY
GO
```

# **B.** Applying multiple transaction log backups

This example applies multiple transaction log backups to the **MyNwind** database.

RESTORE LOG MyNwind FROM MyNwind\_log1 WITH NORECOVERY GO RESTORE LOG MyNwind FROM MyNwind\_log2 WITH NORECOVERY GO RESTORE LOG MyNwind FROM MyNwind\_log3 WITH RECOVERY GO

# See Also

Transaction Log Backups

**RESTORE** 

**Reducing Recovery Times** 

File and Filegroup Backup and Restore

## How to create a differential database backup (Transact-SQL)

#### To create a differential database backup

**IMPORTANT** It is not possible to create a differential database backup unless the database has been backed up first.

- 1. Execute the BACKUP DATABASE statement to create the differential database backup, specifying:
  - The name of the database to back up.
  - The backup device where the database backup will be written.
  - The DIFFERENTIAL clause, to specify that only the parts of the database that have changed after the last database backup was created are backed up.
- 2. Optionally, specify:
  - The INIT clause to overwrite the backup media, and write the backup as the first file on the backup media. If no existing media header exists, one is automatically written.
  - The SKIP and INIT clauses to overwrite the backup media even if there are either backups on the backup media that have not yet expired, or the media name does not match the name on the backup media.
  - The FORMAT clause when using media for the first time to completely initialize the backup media and rewrite any existing media header.

The INIT clause is not required if the FORMAT clause is specified.

**IMPORTANT** Use extreme caution when using the FORMAT or INIT clauses of the BACKUP statement as this will destroy any backups previously stored on the backup media.

## Examples

This example creates a full and a differential database backup for the **MyNwind** database.

```
-- Create a full database backup first.
BACKUP DATABASE MyNwind
TO MyNwind_1
WITH INIT
GO
-- Time elapses.
-- Create a differential database backup, appending the backup
-- to the backup device containing the database backup.
BACKUP DATABASE MyNwind
TO MyNwind_1
WITH DIFFERENTIAL
GO
```

#### See Also

BACKUP

**Differential Database Backups** 

**Reducing Recovery Times** 

### How to restore a differential database backup (Transact-SQL)

#### To restore a differential database backup

- 1. Execute the RESTORE DATABASE statement, specifying the NORECOVERY clause, to restore the database backup preceding the differential database backup. For more information, see <u>How to restore a database backup</u>.
- 2. Execute the RESTORE DATABASE statement to restore the differential database backup, specifying:
  - The name of the database to which the differential database backup will be applied.
  - The backup device where the differential database backup will be restored from.
  - The NORECOVERY clause if you have transaction log backups to apply after the differential database backup is restored, otherwise specify the RECOVERY clause.

## Examples

## A. Restoring a database and differential database backup

This example restores a database and differential database backup of the **MyNwind** database.

-- Assume the database is lost at this point. Now restore the full

-- database. Specify the original full backup and NORECOVERY.

-- NORECOVERY allows subsequent restore operations to proceed. RESTORE DATABASE MyNwind

FROM MyNwind\_1

WITH NORECOVERY

GO

-- Now restore the differential database backup, the second backup on -- the MyNwind\_1 backup device.

**RESTORE DATABASE MyNwind** 

```
FROM MyNwind_1
WITH FILE = 2,
RECOVERY
```

GO

# **B.** Restoring a database, differential database, and transaction log backup

This example restores a database, differential database, and transaction log backup of the **MyNwind** database.

-- Assume the database is lost at this point. Now restore the full

```
-- database. Specify the original full backup and NORECOVERY.
```

-- NORECOVERY allows subsequent restore operations to proceed. RESTORE DATABASE MyNwind

FROM MyNwind\_1

WITH NORECOVERY

GO

```
-- Now restore the differential database backup, the second backup on
```

```
-- the MyNwind_1 backup device.
```

**RESTORE DATABASE MyNwind** 

FROM MyNwind\_1

```
WITH FILE = 2,
```

NORECOVERY

GO

```
-- Now restore each transaction log backup created after
```

```
-- the differential database backup.
```

```
RESTORE LOG MyNwind
```

```
FROM MyNwind_log1
```

```
WITH NORECOVERY
GO
RESTORE LOG MyNwind
FROM MyNwind_log2
WITH RECOVERY
GO
```

### See Also

**Differential Database Backups** 

**RESTORE** 

**Reducing Recovery Times** 

#### How to recover a database without restoring (Transact-SQL)

#### To recover a database without restoring

- Execute the RESTORE DATABASE statement, specifying:
  - The name of the database to be recovered.
  - The RECOVERY clause.

## Examples

This example recovers the **MyNwind** database without restoring from a backup.

-- Restore database using WITH RECOVERY. RESTORE DATABASE MyNwind WITH RECOVERY

### See Also

**Recovering a Database Without Restoring** 

**RESTORE** 

### How to restore to the point of failure (Transact-SQL)

#### To restore to the point of failure

- 1. Execute the BACKUP LOG statement using the NO\_TRUNCATE clause to back up the currently active transaction log.
- 2. Execute the RESTORE DATABASE statement using the NORECOVERY clause to restore the database backup.
- 3. Execute the RESTORE LOG statement using the NORECOVERY clause to apply each transaction log backup.
- 4. Execute the RESTORE LOG statement using the RECOVERY clause to apply the transaction log backup created in Step 1.

## Examples

This example backs up the currently active transaction log of the **MyNwind** database, even though **MyNwind** is inaccessible, and then restores the database to the point of failure using previously created backups:

```
Back up the currently active transaction log.
BACKUP LOG MyNwind
TO MyNwind_log2
WITH NO_TRUNCATE
GO
Restore the database backup.
RESTORE DATABASE MyNwind
FROM MyNwind_1
WITH NORECOVERY
GO
Restore the first transaction log backup.
```

RESTORE LOG MyNwind FROM MyNwind\_log1 WITH NORECOVERY GO -- Restore the final transaction log backup. RESTORE LOG MyNwind FROM MyNwind\_log2 WITH RECOVERY GO

#### See Also

**RESTORE** 

Restoring a Database to a Prior State

## How to set up, maintain, and bring online a standby server (Transact-SQL)

Setting up a standby server generally involves creating a database backup and periodic transaction log backups at the primary server, and then applying those backups, in sequence, to the standby server. The standby server is left in a read-only state between restores. When the standby server needs to be made available for use, any outstanding transaction log backups, including the backup of the active transaction log, from the primary server, are applied to the standby server and the database is recovered.

#### To create backups on the primary server

- 1. Execute the BACKUP DATABASE statement to create the database backup.
- 2. Execute the BACKUP LOG statement to create a transaction log backup.
- 3. Repeat Step 2 for each transaction log you want to create over time.

#### To set up and maintain the standby server

- 1. Execute the RESTORE DATABASE statement using the STANDBY clause to restore the database backup created in Step 1 on the primary server. Specify the name of the undo file that contains the contents of data pages before uncommitted transactions affecting those pages were rolled back.
- 2. Execute the RESTORE LOG statement using the STANDBY clause to apply each transaction log created in Step 2 on the primary server.
- 3. Repeat Step 2 for each transaction log created on the primary server.

#### To bring the standby server online (primary server failed)

- 1. Execute the BACKUP LOG statement using the NO\_TRUNCATE clause to back up the currently active transaction log. This is the last transaction log backup that will be applied to the standby server when the standby server is brought online. For more information, see <u>How to create a backup of the currently active transaction log</u>.
- 2. Execute the RESTORE LOG statement using the STANDBY clause to apply all transaction log backups, including the active transaction log backup created in Step 1, that have not yet been applied to the standby server.
- 3. Execute the RESTORE DATABASE WITH RECOVERY statement to recover the database and bring up the standby server.

## Examples

This example sets up the **MyNwind** database on a standby server. The database can be used in read-only mode between restore operations.

```
-- Restore the initial database backup on the standby server.
USE master
GO
RESTORE DATABASE MyNwind
FROM MyNwind_1
WITH STANDBY = 'c:\undo.ldf'
GO
-- Apply the first transaction log backup.
RESTORE LOG MyNwind
FROM MyNwind_log1
WITH STANDBY = 'c:\undo.ldf'
GO
-- Apply the next transaction log backup.
RESTORE LOG MyNwind
FROM MyNwind_log2
```

```
WITH STANDBY = 'c:\undo.ldf'
```

GO

- -- Repeat for each transaction log backup created on the
- -- primary server.

--

-- Time elapses.....

--

-- The primary server has failed. Back up the

-- active transaction log on the primary server.

```
BACKUP LOG MyNwind
```

TO MyNwind\_log3

WITH NO\_TRUNCATE

GO

-- Apply the final (active) transaction log backup

-- to the standby server. All preceding transaction

-- log backups must have been already applied.

```
RESTORE LOG MyNwind
```

FROM MyNwind\_log3

WITH STANDBY = 'c:\undo.ldf'

GO

-- Recover the database on the standby server,

-- making it available for normal operations.

**RESTORE DATABASE MyNwind** 

```
WITH RECOVERY
GO
```

## See Also

#### **RESTORE**

Restoring a Database to a Prior State

**Using Standby Servers** 

#### How to restore to a point in time (Transact-SQL)

#### To restore to a point in time

- 1. Execute the RESTORE DATABASE statement using the NORECOVERY clause.
- 2. Execute the RESTORE LOG statement to apply each transaction log backup, specifying:
  - The name of the database to which the transaction log will be applied.
  - The backup device from where the transaction log backup will be restored.
  - The RECOVERY and STOPAT clauses. If the transaction log backup does not contain the requested time (for example, if the time specified is beyond the end of the time covered by the transaction log), a warning is generated and the database remains unrecovered.

### Examples

This example restores a database to its state as of 10:00 A.M. on July 1, 1998, and illustrates a restore operation involving multiple logs and multiple backup devices.

-- Restore the database backup. RESTORE DATABASE MyNwind FROM MyNwind\_1, MyNwind\_2 WITH NORECOVERY GO RESTORE LOG MyNwind

```
FROM MyNwind_log1
WITH RECOVERY, STOPAT = 'Jul 1, 1998 10:00 AM'
GO
RESTORE LOG MyNwind
FROM MyNwind_log2
WITH RECOVERY, STOPAT = 'Jul 1, 1998 10:00 AM'
GO
```

#### See Also

**RESTORE** 

Restoring a Database to a Prior State

**RESTORE HEADERONLY** 

#### How to restore the master database (Transact-SQL)

#### To restore the master database

- 1. Start Microsoft<sup>®</sup> SQL Server<sup>™</sup> in single-user mode.
- 2. Execute the RESTORE DATABASE statement to restore the **master** database backup, specifying:
  - The backup device from where the **master** database backup will be restored.

### Examples

This example restores the **master** database backup from tape without using a permanent (named) backup device.

```
USE master
GO
RESTORE DATABASE master
FROM TAPE = '\\.\Tape0'
GO
```

#### See Also

**RESTORE** 

Restoring the master Database from a Current Backup

How to start the default instance of SQL Server in single-user mode (Command Prompt)

## How to back up files and filegroups (Transact-SQL)

#### To back up files and filegroups

- Execute the BACKUP DATABASE statement to create the file and filegroup backup, specifying:
  - The name of the database to back up.
  - The backup device where the database backup will be written.
  - The FILE clause for each file to back up.
  - The FILEGROUP clause for each filegroup to back up.

#### Examples

This example performs a backup operation with files and filegroups for the **MyNwind** database.

```
-- Back up the MyNwind file(s) and filegroup(s)
BACKUP DATABASE MyNwind
FILE = 'MyNwind_data_1',
FILEGROUP = 'new_customers',
FILE = 'MyNwind_data_2',
FILEGROUP = 'first_qtr_sales'
TO MyNwind_1
GO
```

#### See Also

**BACKUP** 

File and Filegroup Backup and Restore

## How to restore files and filegroups (Transact-SQL)

#### To restore files and filegroups

**IMPORTANT** The system administrator restoring the files and filegroups must be the only person currently using the database to be restored.

- 1. Execute the RESTORE DATABASE statement to restore the file and filegroup backup, specifying:
  - The name of the database to restore.
  - The backup device from where the database backup will be restored.
  - The FILE clause for each file to restore.
  - The FILEGROUP clause for each filegroup to restore.
  - The NORECOVERY clause. If the files have not been modified after the backup was created, specify the RECOVERY clause.
- 2. If the files have been modified after the file backup was created, execute the RESTORE LOG statement to apply the transaction log backup, specifying:
  - The name of the database to which the transaction log will be applied.
  - The backup device from where the transaction log backup will be restored.
  - The NORECOVERY clause if you have another transaction

log backup to apply after the current one; otherwise, specify the RECOVERY clause.

The transaction log backups, if applied, must cover the time when the files and filegroups were backed up until the end of log (unless ALL database files are restored).

## Examples

This example restores the files and filegroups for the **MyNwind** database. Two transaction logs will also be applied, to restore the database to the current time.

```
USE master
GO
-- Restore the files and files for MyNwind.
RESTORE DATABASE MyNwind
 FILE = 'MyNwind_data_1',
 FILEGROUP = 'new customers',
 FILE = 'MyNwind_data_2',
 FILEGROUP = 'first_qtr_sales'
 FROM MyNwind_1
 WITH NORECOVERY
GO
-- Apply the first transaction log backup.
RESTORE LOG MyNwind
 FROM MyNwind_log1
 WITH NORECOVERY
GO
-- Apply the last transaction log backup.
RESTORE LOG MyNwind
 FROM MyNwind_log2
 WITH RECOVERY
GO
```

See Also

**RESTORE** 

File and Filegroup Backup and Restore

# How to restore files and filegroups over existing files (Transact-SQL)

#### To restore files and filegroups over existing files

**IMPORTANT** The system administrator restoring the files and filegroups must be the only person currently using the database to be restored.

- 1. Execute the RESTORE DATABASE statement to restore the file and filegroup backup, specifying:
  - The name of the database to restore.
  - The backup device from where the database backup will be restored.
  - The FILE clause for each file to restore.
  - The FILEGROUP clause for each filegroup to restore.
  - The REPLACE clause to specify that each file can be restored over existing files of the same name and location.
  - The NORECOVERY clause. If the files have not been modified after the backup was created, specify the RECOVERY clause.
- 2. If the files have been modified after the file backup was created, execute the RESTORE LOG statement to apply the transaction log backup, specifying:
  - The name of the database to which the transaction log will be applied.

- The backup device from where the transaction log backup will be restored.
- The NORECOVERY clause if you have another transaction log backup to apply after the current one; otherwise, specify the RECOVERY clause.

The transaction log backups, if applied, must cover the time when the files and filegroups were backed up.

## Examples

This example restores the files and filegroups for the **MyNwind** database, and replaces any existing files of the same name. Two transaction logs will also be applied to restore the database to the current time.

```
USE master
GO
-- Restore the files and files groups for MyNwind.
RESTORE DATABASE MyNwind
 FILE = 'MyNwind_data_1',
 FILEGROUP = 'new_customers',
 FILE = 'MyNwind_data_2',
 FILEGROUP = 'first_qtr_sales'
 FROM MyNwind_1
 WITH NORECOVERY,
 REPLACE
GO
-- Apply the first transaction log backup.
RESTORE LOG MyNwind
 FROM MyNwind_log1
 WITH NORECOVERY
GO
-- Apply the last transaction log backup.
RESTORE LOG MyNwind
```

FROM MyNwind\_log2 WITH RECOVERY GO

See Also

**RESTORE** 

Copying Databases

#### How to restore files to a new location (Transact-SQL)

#### To restore files to a new location

**IMPORTANT** The system administrator restoring the files must be the only person currently using the database to be restored.

- 1. Optionally, execute the RESTORE FILELISTONLY statement to determine the number and names of the files in the database backup.
- 2. Execute the RESTORE DATABASE statement to restore the database backup, specifying:
  - The name of the database to restore.
  - The backup device from where the database backup will be restored.
  - The MOVE clause for each file to restore to a new location.
  - The NORECOVERY clause. If the files have not been modified since the backup was created, specify the RECOVERY clause.
- 3. If the files have been modified after the file backup was created, execute the RESTORE LOG statement to apply the transaction log backup, specifying:
  - The name of the database to which the transaction log will be applied.
  - The backup device from where the transaction log backup will be restored.

• The NORECOVERY clause if you have another transaction log backup to apply after the current one; otherwise, specify the RECOVERY clause.

The transaction log backups, if applied, must cover the time when the files and filegroups were backed up.

## Examples

This example restores two of the files for the **MyNwind** database that were originally located on the C:\ drive to new locations on the D: \drive. Two transaction logs will also be applied to restore the database to the current time. The RESTORE FILELISTONLY statement is used to determine the number and logical and physical names of the files in the database being restored.

USE master

GO

-- First determine the number and names of the files in the backup. RESTORE FILELISTONLY

FROM MyNwind\_1

```
-- Restore the files for MyNwind.
```

**RESTORE DATABASE MyNwind** 

```
FROM MyNwind_1
```

WITH NORECOVERY,

```
MOVE 'MyNwind_data_1' TO 'D:\MyData\MyNwind_data_1.mdf',
MOVE 'MyNwind_data_2' TO 'D:\MyData\MyNwind_data_2.ndf'
```

GO

```
-- Apply the first transaction log backup.
```

**RESTORE LOG MyNwind** 

```
FROM MyNwind_log1
```

```
WITH NORECOVERY
```

GO

-- Apply the last transaction log backup.

**RESTORE LOG MyNwind** 

FROM MyNwind\_log2

WITH RECOVERY

GO

## See Also

**RESTORE** 

Copying Databases

#### How to restore a database with a new name (Transact-SQL)

#### To restore a database with a new name

- 1. Optionally, execute the RESTORE FILELISTONLY statement to determine the number and names of the files in the database backup.
- 2. Execute the RESTORE DATABASE statement to restore the database backup, specifying:
  - The new name for the database.
  - The backup device from where the database backup will be restored.
  - The NORECOVERY clause if you have transaction log backups to apply after the file backups are restored. Otherwise, specify the RECOVERY clause.

The transaction log backups, if applied, must cover the time when the files were backed up.

• The MOVE clause for each file to restore to a new location if the file names already exist. For example, creating a copy of an existing database on the same server for testing purposes may be necessary. In this case, the database files for the original database already exist, and so different file names need to be specified when the database copy is created during the restore operation.

### Examples

This example creates a new database called **MyNwind2\_Test**. **MyNwind2\_Test** is a copy of the existing **MyNwind2** database that comprises two files: MyNwind2\_data and MyNwind2\_log. Because the **MyNwind2** database already exists, the files in the backup need to be moved during the restore operation. The RESTORE FILELISTONLY statement is used to determine the number and names of the files in the database being restored.

USE master

GO

-- First determine the number and names of the files in the backup.

-- MyNwind\_2 is the name of the backup device.

**RESTORE FILELISTONLY** 

FROM MyNwind\_2

-- Restore the files for MyNwind2\_Test.

RESTORE DATABASE MyNwind2\_Test

FROM MyNwind\_2

WITH RECOVERY,

MOVE 'MyNwind2\_data' TO 'D:\MyData\MyNwind2\_Test\_data.md MOVE 'MyNwind2\_log' TO 'D:\MyData\MyNwind2\_Test\_log.ldf' GO

#### See Also

#### **RESTORE**

**Copying Databases** 

## **Managing Servers**

Microsoft® SQL Server<sup>™</sup> server management comprises a wide variety of administration tasks, including:

- Registering servers and assigning passwords.
- Reconfiguring network connectivity.
- Configuring linked servers. This allows you to execute distributed queries and distributed transactions on OLE DB data sources across the enterprise.
- Configuring remote servers. This allows you to use one SQL Server installation to execute a stored procedure residing on another SQL Server installation.
- Configuring standby and failover servers.
- Setting server configuration options.
- Managing SQL Server messages.
- Setting the polling intervals.

In most cases, you do not need to reconfigure the server. The default settings for the server components, configured during SQL Server setup, allow you to run SQL Server immediately after it is installed. However, server management is necessary in those situations where you want to add new servers, set up special server configurations, change the network connections, or set server configuration options to improve SQL Server performance.

## How to set up a remote server to allow the use of remote stored procedures (Transact-SQL)

To set up a remote server to allow the use of remote stored procedures

- Run the following code on the first server running Microsoft® SQL Server<sup>™</sup>:
   EXEC sp\_addlinkedserver ServerName1, N'SQL Server'
   EXEC sp\_addlinkedserver ServerName2
   EXEC sp\_configure 'remote access', 1
   RECONFIGURE
   GO
- 2. Stop and restart the first SQL Server.
- 3. Run the following code on the second SQL Server. Make sure you are logging in using SQL Server Authentication.
  - -- The example shows how to set up access for a login 'sa'

-- from ServerName1 on ServerName2.

EXEC sp\_addlinkedserver ServerName2, local

EXEC sp\_addlinkedserver ServerName1

EXEC sp\_configure 'remote access', 1

RECONFIGURE

GO

-- Assumes that the login 'sa' in ServerName2 and ServerNam -- have the same password.

EXEC sp\_addremotelogin ServerName1, sa, sa GO

- 4. Stop and restart the second SQL Server.
- 5. Using the **sa** login, you can now execute a stored procedure on the

second SQL Server from the first SQL Server.

## See Also

sp\_addremotelogin sp\_configure sp\_addlinkedserver RECONFIGURE

#### How to disable a remote server setup (Transact-SQL)

#### To disable a remote server setup

 Run the following code on the second server running Microsoft® SQL Server<sup>™</sup>.
 EXEC sp\_remoteoption ServerName1,sa, sa, trusted, false EXEC sp\_dropremotelogin ServerName1, sa, sa RECONFIGURE GO

EXEC sp\_configure 'remote access', 0 EXEC sp\_dropserver ServerName1 EXEC sp\_dropserver ServerName2 RECONFIGURE GO

- 2. Stop and restart the second SQL Server.
- 3. Run the following code on the first SQL Server: EXEC sp\_configure 'remote access', 0 EXEC sp\_dropserver ServerName2 EXEC sp\_dropserver ServerName1 RECONFIGURE GO
- 4. Stop and restart the first SQL Server.

#### See Also

sp\_configure

<u>sp\_remoteoption</u>

sp\_dropremotelogin

RECONFIGURE

<u>sp\_dropserver</u>

# How to set up and perform a log shipping role change (Transact-SQL)

Log shipping supports the changing of roles, which requires these basic steps:

- 1. Create a Data Transformation Services (DTS) package to copy the current primary server to the current secondary server.
- 2. Create a job to back up **syslogins** from the current primary server, copy the file to a directory on the current secondary server, and then execute the DTS package.
- 3. Perform the role change to set the current secondary server as the current primary server.

Before performing a log shipping role change, a maintenance plan for this log shipping pair must exist on the secondary server. A maintenance plan can be created using the Database Maintenance Plan Wizard, or by adding a server as a secondary server using the **Add Secondary** dialog box found in the user interface of the primary database maintenance plan.

## To create a DTS package to copy the logins from the current primary server to the current secondary server

4. Create a DTS package on the current primary server using DTS Designer.

The package should use the **Transfer Logins Task**, located in the list of tasks in the designer.

- 5. In the **Transfer Logins** dialog box on the **Source** tab, in the **Source** server list, enter the source server (the current primary server).
- 6. Click either **Use Windows Authentication** or **Use SQL Server Authentication**.

- 7. On the **Destination** tab, in the **Destination server** list, enter the destination server (the current secondary server).
- 8. Click either **Use Windows Authentication** or **Use SQL Server Authentication**.
- 9. On the **Logins** tab, click either **All server logins detected at package runtime** or **Logins for selected databases**.
- 10. Save the package.

To create a job to back up syslogins from the current primary server, copy the file to a directory on the current secondary server, and then execute the DTS package

- 1. Click **New Job** to open the **New Job Properties** dialog box on the **General** tab. On the current primary server, create a job owned by **sa** or a login with **sysadmin** rights to both servers.
- 2. On the **Steps** tab, click **New** to open the **New Job Step** dialog box, and then create the following job steps:
  - BCP Out

In the **Type** list, select **Operating System Command** (**CmdExec**). In the **Command** text box, enter the command as follows:

bcp master..syslogins out localpath\syslogins.dat /N /S

Click the **Advanced** tab, and then in the **On success action** list, select **Go to the next step**. In the **On failure action** list, select **Quit the job reporting failure**.

• Copy File

In the **Type** list, select **Transact-SQL Script (T-SQL)**. In the **Database** list, specify **master**. In the **Command** text box,

enter the command as follows:

EXEC xp\_cmdshell 'copy *localpath*\syslogins.dat *dest*.

Click the **Advanced** tab, and then in the **On success action** list, select **Go to the next step**. In the **On failure action** list, select **Quit the job reporting failure**.

• Transfer Logins

In the **Type** list, select **Operating System Command** (**CmdExec**). In the **Command** text box, enter the command as follows:

DTSRun /Scurrent\_primary\_server /Uuser\_nName /Pi

3. In the **New Job Properties** dialog box, click the **Schedules** tab, and then create a job schedule that runs either one time or on a recurrent basis.

It is recommended that the job run as close to the time of role change as possible so that the job obtains the most current login information from the primary server.

## To perform the role change to make the current secondary server the current primary server

You must be a SQL Server administrator to perform a server role change.

 Run sp\_change\_primary\_role on the instance of SQL Server marked as the current primary server. The example shows how to make the primary database stop being the primary database. *current\_primary\_dbname* is the name of the current primary database.
 EXEC sp\_change\_primary\_role

 @db\_name = 'current\_primary\_dbname',
 @backup\_log = 1,
 @terminate = 0,
 @final\_state = 2,
 @access\_level = 1

 2. Run **sp\_change\_secondary\_role** on the instance of SQL Server marked as the current secondary server. The example shows how to make the secondary database the primary database. *current\_secondary\_dbname* is the name of the current secondary database.

```
EXEC sp_change_secondary_role

@db_name = 'current_secondary_dbname',

@do_load = 1,

@force_load = 1,

@final_state = 1,

@access_level = 1,

@terminate = 1,

@stopat = NULL

GO
```

3. Run **sp\_change\_monitor\_role** on the instance of SQL Server marked as the monitor. The example shows how to change the monitor to reflect the new primary database. *new\_source\_directory* is the path to the location where the primary server dumps the transaction logs.

```
EXEC sp_change_monitor_role

@primary_server = 'current_primary_server_name',

@secondary_server = 'current_secondary_server_name',

@database = 'current_secondary_dbname',

@new_source = 'new_source_directory'

GO
```

4. Run **sp\_resolve\_logins** on the instance of SQL Server now marked as the primary server (the former secondary server). You must run the stored procedure from the target database.

The example shows how to resolve the logins on the new primary server against the logins from the former primary server. *destination\_path* is the destination share specified in the Copy File job step. *filename* is the same as specified in the BCP Out job step.

GO

*dbname* is the name of the new primary database.

```
EXEC sp_resolve_logins
@dest_db = 'dbname',
@dest_path = 'destination_path',
@filename = 'filename'
GO
```

The former secondary server is now the current primary server and is ready to assume the function of a primary server. The former primary is no longer part of a log shipping pair. You must add the former primary server as a secondary server to the new primary server to establish a log shipping pair between the two databases.

## See Also

sp\_change\_monitor\_role
sp\_change\_primary\_role
sp\_change\_secondary\_role
sp\_resolve\_logins

## How to set up a Log Shipping Monitor (Transact-SQL)

#### To set up a Log Shipping Monitor on an instance of SQL Server

Execute these stored procedures on the server running the instance of Microsoft® SQL Server<sup>™</sup> 2000 that will monitor log shipping.

- 1. Run **sp\_add\_log\_shipping\_primary** to notify the monitor server which machine will be the primary in the log shipping pair. The output of the stored procedure will be the primary\_id, which will be used by the **sp\_add\_log\_shipping\_secondary** stored procedure.
- 2. Run **sp\_add\_log\_shipping\_secondary** to notify the monitor server which machine will be the secondary in the log shipping pair.

## Examples

This example sets up a log shipping monitor for an existing log shipping pair of the **Northwind** database. You will need to have set up log shipping using the Database Maintenance Plan Wizard prior to setting up this monitor manually. Note that a monitor is created during the wizard setup as well.

EXEC sp\_add\_log\_shipping\_primary @primary\_server\_name = 'MyPrimaryServer', @primary\_database\_name = 'Northwind', @maintenance\_plan\_id = '9B4E380E-11D2-41FC-9BA5-A8EB040/ @backup\_threshold = 15, @threshold\_alert = 14420, @threshold\_alert = 14420, @threshold\_alert\_enabled = 1, @planned\_outage\_start\_time = 0, @planned\_outage\_end\_time = 0, @planned\_outage\_weekday\_mask = 0

EXEC sp\_add\_log\_shipping\_secondary

```
@primary_id = 1,
@secondary_server_name = 'MySecondaryServer',
@secondary_database_name = 'Northwind',
@secondary_plan_id = 'B5C330FF-1081-4FCB-83D0-955DDFB56I
@copy_enabled = 1,
@load_enabled = 1,
@out_of_sync_threshold = 15,
@threshold_alert = 14421,
@threshold_alert = 14421,
@threshold_alert_enabled = 1,
@planned_outage_start_time = 0,
@planned_outage_end_time = 0,
@planned_outage_weekday_mask = 0,
@allow_role_change = 0
GO
```

#### See Also

sp\_add\_log\_shipping\_primary

sp\_add\_log\_shipping\_secondary

## How to remove a log shipping pair from the Log Shipping Monitor (Transact-SQL)

To remove a log shipping pair from a Log Shipping Monitor on an instance of Microsoft® SQL Server<sup>™</sup> 2000

- Run **sp\_delete\_log\_shipping\_monitor\_info** on the monitor server. This informs the monitor server which log shipping pair will be deleted. Note that the actual log shipping pair is not deleted. Only the monitor will be affected by this operation.
- Optionally, run **sp\_delete\_database\_backuphistory** on the primary and secondary servers. This removes backup history information about members of the deleted log shipping pair.

## Examples

This example removes a log shipping pair from a Log Shipping Monitor for an existing log shipping pair of the Northwind database:

```
EXEC sp_delete_log_shipping_monitor_info
@primary_server_name = 'MyPrimaryServer',
@primary_database_name = 'Northwind',
@secondary_server_name = 'MySecondaryServer',
@secondary_database_name = 'Northwind'
GO
```

Optionally, the following stored procedure call can be used to remove backup history information about the deleted members of a log shipping pair. Execute this command on each of the primary and secondary servers:

```
EXEC sp_delete_database_backuphistory 'Northwind' GO
```

sp\_delete\_log\_shipping\_monitor\_info

\*\*\*\*\*

# **Automating Administrative Tasks**

Many of the repetitive tasks performed when administering a Microsoft® SQL Server<sup>™</sup> system can be automated.

Jobs and tasks can be defined to run at specific times or after specific events. These jobs are most often defined using SQL Server Enterprise Manager, but they can also be defined using Transact-SQL statements.

## How to create a job (Transact-SQL)

#### To create a job

- 1. Execute **sp\_add\_job** to create a job.
- 2. Execute **sp\_add\_jobstep** to create one or more job steps.
- 3. Execute **sp\_add\_jobschedule** to create a job schedule.

**Note** It is recommended that you execute **sp\_add\_jobserver** after **sp\_add\_jobstep** for maximum efficiency in communicating job changes to all involved servers.

Because local jobs are cached by the local SQL Server Agent, any modifications implicitly force SQL Server Agent to recache the job. Because SQL Server Agent does not cache the job until **sp\_add\_jobserver** is called, it is more efficient to call **sp\_add\_jobserver** last.

## See Also

**Defining Jobs** 

## How to create a master SQL Server Agent job (Transact-SQL)

#### To create a master SQL Server Agent job

- 1. Execute **sp\_add\_job** to create a job.
- 2. Execute **sp\_add\_jobstep** to create one or more job steps.
- 3. Execute **sp\_add\_jobschedule** to create a job schedule.
- 4. Execute **sp\_add\_jobserver** to specify the target servers on which the job is to run.

**Note** It is recommended that you execute **sp\_add\_jobserver** after **sp\_add\_jobstep** for maximum efficiency in communicating job changes to all involved servers.

Changes to master SQL Server Agent jobs must be propagated to all involved target servers. Because target servers do not initially download the job until **sp\_add\_jobserver** is called, it is recommended that all job steps and job schedules for a particular job be created before executing **sp\_add\_jobserver**. Otherwise, **sp\_post\_msx\_operation** must be subsequently called to request that the target server(s) redownload the modified job.

## See Also

How to create a master SQL Server Agent job (Transact-SQL)

## How to modify a master SQL Server Agent job (Transact-SQL) To change the scheduling details for a job definition

• Execute **sp\_update\_jobschedule**.

To add, change, or remove steps from a job by working with the job steps

- 1. Execute **sp\_add\_jobstep** to add new job steps.
- 2. Execute **sp\_update\_jobstep** to change pre-existing job steps.
- 3. Execute **sp\_delete\_jobstep** to delete a pre-existing job.

#### To modify the target server(s) associated with a job

- 1. Execute **sp\_delete\_jobserver** to delete a server currently associated with a job.
- 2. Execute **sp\_add\_jobserver** to associate a server with the current job.

**Note** A master SQL Server Agent job cannot be targeted at both local and remote servers.

### See Also

#### **Creating Jobs**

## How to create an operator (Transact-SQL)

#### To create an operator for a local job

• Execute **sp\_add\_operator**.

#### To create an operator for a master SQL Server Agent

- 1. Execute **sp\_add\_operator** to specify the master SQL Server Agent operator.
- 2. Execute **sp\_add\_targetsvrgrp\_member** to add the specified target server to the target server group
- 3. Execute **sp\_msx\_enlist** to enlist the target server in the job.
- 4. Execute the steps in How to create a master SQL Server Agent job (Transact-SQL) to create a master SQL Server Agent job.

#### See Also

**Defining Operators** 

## How to modify an operator (Transact-SQL)

#### To modify a local operator

• Execute **sp\_update\_operator**.

#### To modify an operator for a master SQL Server Agent

- 1. Execute **sp\_msx\_defect** to remove the target server from the master SQL Server Agent.
- 2. Execute **sp\_update\_operator** to change the operator.

## See Also

Modifying and Viewing Operators

# **Monitoring Server Performance and Activity**

There are a variety of tools and techniques that can be used to monitor Microsoft® SQL Server<sup>TM</sup> 2000. The general reasons for monitoring SQL Server are:

- Determining if performance improvements can be made.
- Determining user activity to find out what queries users are issuing and who is connecting to SQL Server.
- Troubleshooting problems.
- Testing applications.

## How to create a trace (Transact-SQL)

#### To create a trace

- 1. Execute **sp\_trace\_create** with the required parameters to create a new trace. The new trace will be in a stopped state (*status* is 0).
- 2. Execute **sp\_trace\_setevent** with the required parameters to select the events and columns to trace.
- 3. Optionally, execute **sp\_trace\_setfilter** to set any or a combination of filters.

**sp\_trace\_setevent** and **sp\_trace\_setfilter** can be executed only on existing traces that are stopped.

**IMPORTANT** Unlike regular stored procedures, parameters of all SQL Profiler stored procedures (**sp\_trace\_***xx*) are strictly typed and do not support automatic data type conversion. If these parameters are not called with the correct input parameter data types, as specified in the argument description, the stored procedure will return an error.

#### See Also

Creating and Managing Traces and Templates

<u>sp\_trace\_create</u>

<u>sp\_trace\_setevent</u>

## How to set a trace filter (Transact-SQL)

#### To set a trace filter

- If the trace is already running, execute **sp\_trace\_setstatus** specifying *@***status** = 0 to stop the trace.
- 2. Execute **sp\_trace\_setfilter** to configure the type of information to retrieve for the event being traced.

**IMPORTANT** Unlike regular stored procedures, parameters of all SQL Profiler stored procedures (**sp\_trace\_***xx*) are strictly typed and do not support automatic data type conversion. If these parameters are not called with the correct input parameter data types, as specified in the argument description, the stored procedure will return an error.

#### See Also

Creating and Managing Traces and Templates

**Limiting Traces** 

sp\_trace\_setfilter

<u>sp\_trace\_setstatus</u>

System Stored Procedures (SQL Profiler Procedures)

## How to modify an existing trace (Transact-SQL)

#### To modify an existing trace

- If the trace is already running, execute **sp\_trace\_setstatus** specifying *@***status** = 0 to stop the trace.
- 2. To modify trace events, execute **sp\_trace\_setevent**, specifying the changes through the parameters. Listed in order, they are:
  - @traceid (Trace ID)
  - @eventid (Event ID)
  - @columnid (Column ID)
  - @on (ON)

When modifying the *@on* parameter, keep in mind its interaction with the *@columnid* parameter:

ON	Column ID	Result
ON (1)	NULL	Event is turned on.
		All columns are cleared.
	NOT NULL	Column is turned on for the specified
		event.
OFF (0)	NULL	Event is turned off.
		All columns are cleared.
	NOT NULL	Column is turned off for the
		specified event.

**IMPORTANT** Unlike regular stored procedures, parameters of all SQL Profiler stored procedures (**sp\_trace\_***xx*) are strictly typed and do not

support automatic data type conversion. If these parameters are not called with the correct input parameter data types, as specified in the argument description, the stored procedure will return an error.

## See Also

Creating and Managing Traces and Templates

**Modifying Templates** 

<u>sp\_trace\_setevent</u>

System Stored Procedures (SQL Profiler Procedures)

## How to view a saved trace (Transact-SQL)

#### To view a specific trace

• Execute **fn\_trace\_getinfo** specifying the ID of the trace on which information is needed. This function will return a table listing the trace, trace property, and information about the property.

Invoke the function this way:

SELECT \*
FROM ::fn\_trace\_getinfo(trace\_id)

#### To view all existing traces

• Execute **fn\_trace\_getinfo** specifying "0" or the term "default". This function will return a table listing all the trace, their properties, and information about these properties.

Invoke the function this way:

SELECT \*
FROM ::fn\_trace\_getinfo(default)

#### See Also

Creating and Managing Traces and Templates

fn\_trace\_getinfo

Viewing and Analyzing Traces

## How to view filter information (Transact-SQL)

#### To view filter information

• Execute **fn\_trace\_getfilterinfo** specifying the ID of the trace on which filter information is needed. This function will return a table listing the filters, the column on which the filters are applied, and the value on which the filter is applied.

Invoke the function this way:

SELECT \*
FROM ::fn\_trace\_getfilterinfo(trace\_id)

## See Also

**Creating and Managing Traces and Templates** 

fn\_trace\_getfilterinfo

System Stored Procedures (SQL Profiler Procedures)

Viewing and Analyzing Traces

## How to delete a trace (Transact-SQL)

#### To delete a trace

- 1. Execute **sp\_trace\_setstatus** specifying **@status** = 0 to stop the trace.
- 2. Execute **sp\_trace\_setstatus** specifying **@status** = 2 to close the trace and delete its information from the server.

**Note** A trace must be stopped first before it can be closed.

## See Also

Creating and Managing Traces and Templates

**Deleting** Traces

<u>sp\_trace\_setstatus</u>

System Stored Procedures (SQL Profiler Procedures)

# **Integrating SQL Server with Other Tools**

Microsoft® SQL Server<sup>™</sup> applications can reference Automation objects in Transact-SQL statements. SQL Server can also use MAPI-compliant e-mail systems to send and receive e-mails.

## How to create an OLE Automation object (Transact-SQL)

#### To create an OLE Automation object

- 1. Call **sp\_OACreate** to create the object.
- 2. Use the object.
  - Call **sp\_OAGetProperty** to get a property value.
  - Call **sp\_OASetProperty** to set a property to a new value.
  - Call **sp\_OAMethod** to call a method.
  - Call **sp\_OAGetErrorInfo** to get the most recent error information.
- 3. Call **sp\_OADestroy** to destroy the object.

**Note** All of these steps must be performed within a single Transact-SQL statement batch. All created OLE objects are destroyed automatically at the end of each statement batch.

## See Also

System Stored Procedures

Data Type Conversions Using OLE Automation Stored Procedures

How to debug a custom OLE Automation server (Transact-SQL)

**OLE Automation Sample Script** 

## How to debug a custom OLE Automation server (Transact-SQL)

You can debug a custom OLE Automation server created by using 32-bit Microsoft® Visual Basic® version 4.0. To do this, Visual Basic must be installed on the Microsoft SQL Server<sup>™</sup> computer, and SQL Server must be running under the same Microsoft Windows NT® user account as Visual Basic. SQL Server must be started from the command prompt and independently of the Windows NT Service Control Manager (by using the **sqlservr** /**c** command), or the SQL Server service must be started under the same Windows NT user account used to log on to the system.

#### To debug a custom OLE Automation server

- 1. Load your custom OLE Automation server project into Visual Basic.
- 2. Set breakpoint(s) on the desired lines of source code.
- 3. On the **Run** menu, click **Start With Full Compile**.

This registers and runs your custom OLE Automation server.

4. Use the OLE Automation stored procedures to call the OLE objects exposed by your custom OLE Automation server.

When a breakpoint is hit, the Visual Basic debugger is activated.

For more information, see your documentation for Visual Basic.

A custom, in-process OLE server, created using 32-bit Visual Basic 4.0, must have an error handler (specified with the **On Error GoTo** statement) for the **Class\_Initialize** and **Class\_Terminate** subroutines. The error handlers will prevent unhandled errors from occurring in the subroutines. Unhandled errors in the **Class\_Initialize** and **Class\_Terminate** subroutines can cause unpredictable SQL Server problems, such as a SQL Server access violation. Error handlers for other subroutines are also recommended.

#### See Also

System Stored Procedures

How to create an OLE Automation object (Transact-SQL)

Data Type Conversions Using OLE Automation Stored Procedures

OLE Automation Sample Script

## How to use SQL Mail (Transact-SQL)

SQL Mail uses several extended stored procedures that are necessary for mail enabling. These extended stored procedures are included in a dynamic-link library, SQLMAP70.DLL, which is installed with Microsoft® SQL Server<sup>™</sup> 2000.

#### To process e-mail messages manually

- 1. In SQL Query Analyzer, start a SQL Server Mail client session by executing **xp\_startmail**.
- 2. To find the ID of the next unread message in the mail box, execute **xp\_findnextmsg**.
- 3. To read a message or attachment, execute **xp\_readmail** (using a specific message ID), and use the output variable in a SELECT statement to display the message in the result pane.
- 4. To delete a message, execute **xp\_deletemail** (using a specific message ID).
- 5. To send a message or a query result set to specified recipients, execute **xp\_sendmail** (with the query in the message body).
- 6. Stop the SQL Server Mail client session by executing **xp\_stopmail**.

#### To process multiple e-mail messages as a scheduled job

- 1. In SQL Query Analyzer, start a SQL Server Mail client session by executing **xp\_startmail**.
- 2. Execute **sp\_processmail** to find, read, respond to, and delete multiple

messages.

3. Stop the SQL Server Mail client session by executing **xp\_stopmail**.

## See Also

- sp\_processmail
- <u>xp\_sendmail</u>
- <u>xp\_deletemail</u>
- <u>xp\_startmail</u>
- xp\_findnextmsg
- <u>xp\_stopmail</u>
- <u>xp\_readmail</u>

# Replication

Microsoft® SQL Server<sup>™</sup> 2000 replication is the process of copying and distributing data and database objects from one database to another and then synchronizing between databases for consistency.

Using replication, you can distribute data to different locations, to remote or mobile users over a local area network, using a dial-up connection, and over the Internet. Replication also allows you to enhance application performance, physically separate data based on how it is used (for example, to separate online transaction processing (OLTP) and decision support systems), or distribute database processing across multiple servers.

# **Replication Types**

Microsoft<sup>®</sup> SQL Server<sup>™</sup> 2000 provides the following types of replication that you can use in your distributed applications:

- Snapshot replication
- Transactional replication
- Merge replication

Each type provides different capabilities depending on your application and different levels of ACID properties of transactions and site autonomy. For example, merge replication allows users to work and update data autonomously, although ACID properties are not assured. Instead, when servers are reconnected, all sites in the replication topology converge to the same data values. Transactional replication maintains transactional consistency, but Subscriber sites are not as autonomous as they are in merge replication because Publishers and Subscribers generally must be connected reliably and continuously for updates to be propagated to Subscribers.

It is common for the same application to use multiple replication types and options. Some of the data in the application may not require any updates at Subscribers, some sets of data may require updates infrequently, with updates made at only one or a few servers, while other sets of data may need to be updated daily at multiple servers.

Which type of replication you choose for your application depends on your requirements based on distributed data factors, whether or not data will need to be updated at the Subscriber, your replication environment, and the needs and requirements of the data that will be replicated. For more information, see <u>Planning for Replication</u>.

Each type of replication begins with generating and applying the snapshot at the Subscriber, so it is important to understand snapshot replication in addition to any other type of replication and options you choose.

# How to set row- or column-level tracking for an article (Transact-SQL)

- If you are adding a merge article to a publication, set the *@column\_tracking* parameter of the replication stored procedure **sp\_addmergearticle** to **true** for column-level tracking or to **false** for row-level tracking.
- If you are changing the properties of an existing inactive merge article in a publication, set the *@property* = parameter of the replication stored procedure **sp\_changemergearticle** to **column\_tracking**, and then set the *@value* = parameter to **true** for column-level tracking or to **false** for row-level tracking.

If this property is changed after the publication has active subscriptions, the current snapshot will become obsolete and existing subscriptions will be marked for reinitialization.

## How to choose a resolver (Transact-SQL)

- If you are using stored procedures to create a publication and want to specify the resolver, set the *@article\_resolver* = parameter of the replication stored procedure **sp\_addmergearticle** to the name of the custom resolver. If the custom resolver is a stored procedure, also set the *@resolver\_info* = parameter to the name of the stored procedure.
- If you are changing the resolver properties of an existing merge article in a publication, set the *@property* = parameter of the replication stored procedure **sp\_changemergearticle** to **article\_resolver**, and then set the *@value* = parameter to the name of the custom resolver. If the custom resolver is a stored procedure, execute a second **sp\_changemergearticle** statement, set the *@property* = parameter to **resolver\_info**, and then set the *@value* = parameter to the name of the

stored procedure.

# **Implementing Replication (Transact-SQL)**

Whether you are using snapshot replication, transactional replication, or merge replication, the following stages will help you implement replication.

Stage	Tasks
Configuring Replication	Identify the Publisher, Distributor, and Subscribers in your topology. Use SQL Server Enterprise Manager, SQL-DMO, scripts, or Transact-SQL system stored procedures to configure the Publisher, create a distribution database, and enable Subscribers.
Publishing Data and Database Objects	Create the publication and define the data and database object articles in the publication, and apply any necessary filters to data that will be published.
Subscribing to Publications	Create push, pull, or anonymous subscriptions to indicate what publications need to be propagated to individual Subscribers and when.
Applying the Initial Snapshot	Indicate where to save snapshot files, whether they are compressed, and scripts to run before or after applying the initial snapshot. Specify to have the Snapshot Agent apply the
	snapshot at the Subscriber immediately after creating a subscription or at a specified time.
	Apply the snapshot manually by saving it to a network location or to removable media that can be transported to the Subscriber, and then applying the Snapshot files manually at the Subscriber.
Synchronizing Data	Synchronizing data occurs when the Snapshot, Log Reader, or Merge Agent runs and updates are propagated between Publisher and Subscribers.

For snapshot replication, the snapshot will be reapplied at the Subscriber.
For transactional replication, updates will be propagated to Subscribers.
If using updatable subscriptions with either snapshot replication or transactional replication, data will be propagated from the Subscriber to the Publisher and to other Subscribers.
For merge replication, data is synchronized during the merge process when data changes at all servers are converged and conflicts, if any, are detected and resolved.

# How to Configure Publishing and Distribution (Transact-SQL)

- 1. Execute **sp\_adddistributor** at the server that will be the Distributor.
- 2. Execute **sp\_adddistributiondb** at the Distributor to create a new distribution database.
- 3. Execute **sp\_adddistpublisher** at each server that will be a Publisher using the Distributor.

On the **master** database on the Publisher, execute **sp\_replicationdboption** for each database that will be a publication database.

# How to Modify Publisher and Distributor Properties (Transact-SQL)

#### To view Distributor properties

- 1. Execute **sp\_helpdistributor** to list information about the Distributor, distribution database, working directory, and SQL Server Agent user account.
- 2. Execute **sp\_helpdistributiondb** to return properties of the specified distribution database.

#### To modify a Distributor

- 1. Execute **sp\_changedistributor\_property** to modify Distributor properties.
- 2. Execute **sp\_changedistributiondb** to modify distribution database properties.

#### To add a password to a Distributor

• Execute **sp\_add\_distributor** specifying the *password* parameter.

#### To change a password on a Distributor

• Execute **sp\_changedistributor\_password**.

#### To create a new distribution database

• Execute **sp\_adddistributiondb** to create a new distribution database and install the distribution schema.

#### To add a Publisher to a distribution database

• Execute **sp\_adddistpublisher** *n* times to define each Publisher that uses the Distributor.

### To remove a Publisher from a distribution database

• Execute **sp\_dropdistpublisher** to drop a Publisher that is also the Distributor.

## How To Disable Publishing and Distribution (Transact-SQL)

#### To disable a Distributor (Transact-SQL)

- 1. Execute **sp\_dropdistpublisher** to drop a Publisher that is also the Distributor.
- 2. Execute **sp\_dropdistributiondb** to delete the distribution database.
- 3. Execute **sp\_dropdistributor** to remove the Distributor designation from the server.

To delete a distribution database (Transact-SQL)

• Execute **sp\_dropdistributiondb** to delete a distribution database.

To manually remove replication on a Publisher without connecting to the Distributor (Transact-SQL)

• Execute **sp\_dropdistributor** with **@no\_checks=1** and **@ignore\_distributor=1**.

# How to Create Publications and Define Articles (Transact-SQL)

#### To create a snapshot or transactional publication

- 1. Execute **sp\_replicationdboption** to enable publication of the current database.
- 2. Execute **sp\_addpublication** with **repl\_freq** set to snapshot to define the publication.
- 3. Execute **sp\_addpublication\_snapshot** to create a Snapshot Agent, set the publication **agent\_id**, and place the schema and data into the replication working directory.
- 4. Execute **sp\_addarticle** *n* times to define each article in the publication.

#### To define an article for a snapshot or transactional publication

- 1. Execute **sp\_addarticle** to define an article.
- 2. Execute **sp\_articlefilter** to filter a table horizontally.
- 3. Execute **sp\_articlecolumn** to filter a table vertically.
- 4. Execute **sp\_articleview** to create the synchronization object for an article when a table is filtered vertically or horizontally.

#### To create a merge publication

1. Execute **sp\_replicationdboption** to enable publication of the current database.

- 2. Execute **sp\_addmergepublication** to define the publication.
- 3. Execute **sp\_addpublication\_snapshot** to create a Snapshot Agent and place the schema and data into the replication working directory.
- 4. Execute **sp\_addmergearticle** *n* times to define each article in the publication.

#### To define a merge article

- 1. Execute **sp\_addmergearticle** to define an article.
- 2. Execute **sp\_addmergefilter** to create a partitioned publication.

# How to Modify Publications and Articles (Transact-SQL)

#### To view snapshot or transactional publication properties

• Execute **sp\_helppublication** to display information about a publication.

#### To view merge publication properties

• Execute **sp\_helpmergepublication** to display information about a publication.

#### To modify publication properties

• Execute **sp\_changepublication** to modify the properties of a publication.

#### To modify merge publication properties

• Execute **sp\_changemergepublication** to modify the properties of a publication.

#### To view article properties for a snapshot or transactional publication

- 1. Execute **sp\_helparticle** to display information about an article.
- 2. Execute **sp\_helparticlecolumns** to display all columns in the table underlying an article.

#### To view merge article properties

- 1. Execute **sp\_helpmergearticle** to display information about a merge article.
- 2. Execute **sp\_helpmergefilter** to display information about merge filters.

#### To modify article properties for a snapshot or transactional publication

• Execute **sp\_changearticle** to change the properties of an article.

## To modify merge article properties

- 1. Execute **sp\_changemergearticle** to change the properties of an article.
- 2. Execute **sp\_changemergefilter** to change the properties of a filter.

# How to Delete Publications and Articles (Transact-SQL)

#### To delete a snapshot or transactional publication

- 1. Execute **sp\_dropsubscription** to delete all snapshot subscriptions.
- 2. Execute **sp\_droppublication** to delete the publication and all of its articles.
- 3. Execute **sp\_replicationdboption** to disable replication of the current database.

**Note** Do not call **sp\_replicationdboption** to disable publication of the current database if you are deleting only one publication and you still want to publish from the database.

#### To delete an article for a snapshot or transactional publication

• Execute **sp\_droparticle** to delete an article from a publication.

#### To delete a merge publication

- 1. Execute **sp\_dropmergesubscription** to delete all merge subscriptions.
- 2. Execute **sp\_mergesubscription\_cleanup** to remove merge configuration for all merge articles in the subscription database.
- 3. Execute **sp\_dropmergepublication** to delete the publication and all of its articles.
- 4. Execute **sp\_replicationdboption** to disable replication of the current database.

**Note** Do not call **sp\_replicationdboption** to disable publication of the current database if you are deleting only one publication and you want to publish from

the database.

### To delete a merge article

• Execute **sp\_dropmergearticle** to delete an article from a publication.

# How to Create a Push Subscription (Transact-SQL)

#### To add a push subscription for a snapshot publication (Transact-SQL)

- 1. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 2. Execute **sp\_addpublication** with **allow\_push** set to TRUE to enable push subscriptions.
- 3. Execute **sp\_addsubscription** to create the subscription.

#### To create a push subscription (transactional)

- 1. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 2. Execute **sp\_addpublication** with *allow\_push* set to true to enable push subscriptions.
- 3. Execute **sp\_addsubscription** to create the subscription.

#### To add a push subscription (merge)

- 1. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 2. Execute **sp\_addmergepublication** with *allow\_push* set to true to enable push subscriptions.
- 3. Execute **sp\_addmergesubscription** to create the subscription.

# How to Modify a Push Subscription (Transact-SQL)

#### To view push subscription properties for a snapshot publication

- 1. Execute **sp\_helpsubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of subscriptions.
- 2. Execute **sp\_helpsubscriberinfo** to display information about a Subscriber.

#### To modify push subscription properties for a snapshot publication

- 1. Execute **sp\_changesubscriber** to change Subscriber options.
- 2. Execute **sp\_changesubstatus** to change Subscriber status.

#### To view push subscription properties for a transactional publication

- 1. Execute **sp\_helpsubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of subscriptions.
- 2. Execute **sp\_helpsubscriberinfo** to display information about a Subscriberexecute.

#### To modify push subscription properties for a transactional publication

- 1. Execute **sp\_changesubscriber** to change Subscriber options.
- 2. Execute **sp\_changesubstatus** to change Subscriber status.

#### To view push subscription properties for a merge publication

• Execute **sp\_helpmergesubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of

subscriptions.

# To modify push subscription properties for a merge publication

• Execute **sp\_changemergesubscription** to change Subscriber options.

# How to Delete a Push Subscription (Transact-SQL)

#### To delete a push subscription to a snapshot publication

- 1. Execute **sp\_dropsubscription** to delete the subscription.
- 2. Execute **sp\_dropsubscriber** to remove the registration entry of the Subscriber.

**Note** It is not necessary to drop a Subscriber unless you are dropping the last publication to which it subscribes.

#### To delete a push subscription to a transactional publication

- 1. Execute **sp\_dropsubscription** to delete the subscription.
- 2. Execute **sp\_dropsubscriber** to remove the registration entry of a Subscriber.

**Note** It is not necessary to drop a Subscriber unless you are dropping the last publication to which it subscribes.

#### To delete a push subscription to a merge publication

• Execute **sp\_dropmergesubscription** to delete the subscription.

# How to Create a Pull Subscription (Transact-SQL)

#### To add a pull subscription to a snapshot publication

- 1. Execute **sp\_addpublication** with *allow\_pull* set to true to enable pull subscriptions at the Publisher.
- 2. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 3. Execute **sp\_addsubscription** to create the subscription at the Publisher.
- 4. Execute **sp\_addpullsubscription** to create the pull subscription at the Subscriber.
- 5. Execute **sp\_addpullsubscription\_agent** to create a scheduled job for the Distribution Agent at the Subscriber.

#### To create a pull subscription (transactional)

- 1. Execute **sp\_addpublication** with *allow\_pull* set to TRUE to enable pull subscriptions at the Publisher.
- 2. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 3. Execute **sp\_addsubscription** to create the subscription at the Publisher.
- 4. Execute **sp\_addpullsubscription** to create the pull subscription at the Subscriber.
- 5. Execute **sp\_addpullsubscription\_agent** to create a scheduled job for

the Distribution Agent at the Subscriber.

#### To add a pull subscription (merge)

- 1. Execute **sp\_addmergepublication** with *allow\_pull* set to TRUE to enable pull subscriptions at the Publisher.
- 2. Execute **sp\_addsubscriber** to register the Subscriber at the Publisher.
- 3. Execute **sp\_addmergesubscription** to create the subscription at the Publisher.
- 4. Execute **sp\_addmergepullsubscription** to create the subscription at the Subscriber.
- 5. Execute **sp\_addmergepullsubscription\_agent** to create a scheduled job for the Distribution Agent at the Subscriber.

# How to View or Modify Pull or Anonymous Subscriptions (Transact-SQL)

# To view pull and anonymous subscription properties for a snapshot publication

- 1. Execute **sp\_helpsubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of subscriptions.
- 2. Execute **sp\_helppullsubscription** to display information about one or more subscriptions at the Subscriber.
- 3. Execute **sp\_helpsubscriberinfo** to display information about the Subscriber.

# To modify pull and anonymous subscription properties for a snapshot publication

- 1. Execute **sp\_changesubscriber** to change Subscriber options.
- 2. Execute **sp\_changesubstatus** to change Subscriber status.

#### To view pull and anonymous subscription properties (transactional)

- 1. Execute **sp\_helpsubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of subscriptions.
- 2. Execute **sp\_helppullsubscription** to display information about one or more subscriptions at the Subscriber.
- 3. Execute **sp\_helpsubscriberinfo** to display information about the Subscriber.

To modify pull and anonymous subscription properties (transactional)

- 1. Execute **sp\_changesubscriber** to change Subscriber options.
- 2. Execute **sp\_changesubstatus** to change Subscriber status.

#### To view pull and anonymous subscription properties (merge)

• Execute **sp\_helpmergepullsubscription** to list subscription information associated with a particular publication, article, Subscriber, or set of subscriptions.

#### To modify pull and anonymous subscription properties (merge)

• Execute **sp\_changemergepullsubscription** to change Subscriber options.

# How to Delete a Pull Subscription (Transact-SQL)

#### To delete a pull subscription to a snapshot publication

- 1. Execute **sp\_dropsubscription** to delete the subscription.
- 2. Execute **sp\_dropsubscriber** to remove the registration entry of the Subscriber.

#### To delete a pull subscription (transactional)

- 1. Execute **sp\_dropsubscription** to delete the subscription.
- 2. Execute **sp\_dropsubscriber** to remove the registration entry of the Subscriber.
- 3. Execute **sp\_droppullsubscription** at the Subscriber.

#### To delete a pull subscription (merge)

• Execute **sp\_dropmergepullsubscription** to delete the subscription.

# How to Create an Anonymous Subscription (Transact-SQL)

To add an anonymous subscription to a snapshot publication (Transact-SQL)

- 1. Execute **sp\_addpublication** with *allow\_pull*, *allow\_anonymous*, and *immediate\_sync* set to TRUE to enable anonymous subscriptions.
- 2. Execute **sp\_addpullsubscription** to create the anonymous subscription at the Subscriber.
- 3. Execute **sp\_addpullsubscription\_agent** to create a scheduled job for the Distribution Agent at the Subscriber.

#### To add an anonymous subscription (transactional)

- 1. Execute **sp\_addpublication** with *allow\_pull*, *allow\_anonymous*, and *immediate\_sync* set to TRUE to enable anonymous subscriptions.
- 2. Execute **sp\_addpullsubscription** to create the anonymous subscription at the Subscriber.
- 3. Execute **sp\_addpullsubscription\_agent** to create a scheduled job for the Distribution Agent at the Subscriber.

#### To add an anonymous subscription to a merge publication

- 1. Execute **sp\_addmergepublication** with *allow\_pull* and *allow\_anonymous* set to TRUE to enable anonymous subscriptions at the Publisher.
- 2. Execute **sp\_addmergepullsubscription** to create the anonymous subscription at the Subscriber.

3. Execute **sp\_addmergepullsubscription\_agent** to create a scheduled job for the anonymous Merge Agent at the Subscriber.

# How to Delete an Anonymous Subscription (Transact-SQL)

#### To disable snapshot publications that allow anonymous subscriptions

- 1. Execute **sp\_droparticle** *n* times to delete each article in the publication.
- 2. Execute **sp\_droppublication** to delete the publication.
- 3. Execute **sp\_replicationdboption** to disable replication of the current database.

**Note** Anonymous subscriptions are unknown to the Publisher. The preceding steps disable all anonymous subscriptions to a publication by dropping the publication.

#### To disable transactional publications that allow anonymous subscriptions

- 1. Execute **sp\_droparticle** *n* times to delete each article in the publication.
- 2. Execute **sp\_droppublication** to delete the publication.
- 3. Execute **sp\_replicationdboption** to disable replication of the current database.

**Note** Anonymous subscriptions are unknown to the Publisher. The preceding steps disable all anonymous subscriptions to a publication by dropping the publication.

#### To disable merge publications that allow anonymous subscriptions

1. Execute **sp\_dropmergearticle** *n* times to delete each article in the publication.

- 2. Execute **sp\_dropmergepublication** to delete the publication.
- 3. Execute **sp\_replicationdboption** to disable replication of the current database.

**Note** Anonymous subscriptions are unknown to the Publisher. The preceding steps disable all anonymous subscriptions to a publication by dropping the publication.

# How to Browse and Copy Snapshot Files (Transact-SQL)

#### To browse snapshot files

- For transactional publications, execute the **sp\_browsesnapshotfolder** Transact-SQL system stored procedure at the Publisher.
- For merge publications, execute the **sp\_browsemergesnapshotfolder** Transact-SQL system stored procedure at the Publisher.

#### To copy snapshot files

- For transactional publications, execute the **sp\_copysnapshot** Transact-SQL system stored procedure at the Publisher.
- For merge publications, execute the **sp\_copymergesnapshot** Transact-SQL system stored procedure at the Publisher.

# How to Apply Schema Changes on Publication Databases (Transact-SQL)

#### To add columns to an article

• Execute the **sp\_repladdcolumn** stored procedure and set the following parameters.

#### Parameter

**Function** @*source\_object*Names the table to which the column will be added. @*column*Names the column to be added. @*typetext*Defines the column (data type information, default value, and so on.). In the SQL for the column definition, you must either specify a default value or allow NULL values.

For information about the syntax required to define the column, see <u>ALTER</u> <u>TABLE</u>.

@publication\_to\_addLists the names of the publications to which you will add the column; you can also use the values **all** or **none**.

*@force\_invalidate\_snapshot*When set equal to 0, current snapshot with previous schema information is still available in case it is needed. This parameter affects only publications created with the **immediate\_sync** option.

@force\_reinit\_subscriptionWhen set equal to 1, schema changes commands will not be propagated to Subscribers. All subscriptions affected by the schema change will be reintialized except for nosync subscriptions, for which no action is taken.

#### To drop columns from an article

• Execute the **sp\_repldropcolumn** stored procedure and set the following parameters.

#### Parameter

**Function** @*source\_object*Names the table from which the column will be dropped. @*column*Names the column to be dropped.

@force\_invalidate\_snapshotWhen set equal to 0, current snapshot with previous

schema information is still available in case it is needed. This parameter affects only publications created with the **immediate\_sync** option.

@force\_reinit\_subscriptionWhen set equal to 1, schema changes commands will not be propagated to Subscribers. All subscriptions affected by the schema change will be reintialized except for nosync subscriptions, for which no action is taken.

## How to Publish Data Over the Internet (Transact-SQL)

#### To publish over the Internet using snapshot replication (Transact-SQL)

• When defining the publication, execute **sp\_addpublication** with *enabled\_for\_internet* set to TRUE to enable Internet subscriptions.

#### To publish on the Internet using transactional replication (Transact-SQL)

• When defining the publication, execute **sp\_addpublication** with *enabled\_for\_internet* set to TRUE to enable Internet subscriptions.

#### To publish on the Internet using merge replication

• When defining the publication, execute **sp\_addmergepublication** with *enabled\_for\_internet* set to TRUE to enable Internet subscriptions.

# **Replication Options (Transact-SQL)**

Replication Options allow you to configure replication in a manner best suited to your application and environment.

Option	Type of Replication	Benefits
Filtering Published Data	Snapshot Replication Transactional Replication Merge Replication	<ul> <li>Filters allow you to create vertical and/or horizontal partitions of data that can be published as part of replication. By distributing partitions of data to different Subscribers, you can: <ul> <li>Minimize the amount of data sent over the network.</li> </ul> </li> <li>Reduce the amount of storage space required at the Subscriber.</li> <li>Customize publications and applications based on individual Subscriber requirements.</li> <li>Reduce conflicts because the different data partitions can be sent to different Subscribers.</li> </ul>
Updatable Subscriptions (Immediate Updating, Queued Updating)	Snapshot Replication Transactional Replication	Immediate updating and queued updating options allow users to update data at the Subscriber and either propagate those updates to the Publisher immediately or store the updates in a queue. Updatable subscriptions are best for replication topologies where replicated

		data is mostly read, and occasionally updated at the Subscriber when Publisher, Distributor, and Subscriber are connected most of the time and when conflicts caused my multiple users updating the same data are infrequent.
Transforming Published Data	Snapshot Replication Transactional Replication	<ul> <li>You can leverage the data movement, transformation mapping and filtering capabilities of Data Transformation</li> <li>Services (DTS) during replication. With transformable subscriptions, you can: <ul> <li>Create custom partitions for snapshot and transactional publications.</li> </ul> </li> <li>Transform the data as it is being published with data type mappings (for example, integer to real data type), column manipulations (for example, concatenating first name and last name columns into one), string manipulations, and functions.</li> </ul>
Alternate Synchronization Partners	Merge Replication	Alternate synchronization partners allow Subscribers to merge publications to synchronize data with servers other than the Publisher at which the subscription originated. This allows the Subscriber to synchronize data when the original Publisher is unavailable, and is also useful for mobile Subscribers that may have access to a faster or more reliable network connection with an alternate server.
Optimizing	Merge	By optimizing synchronization during

Synchronization	Replication	merge replication, you can store more information at the Publisher instead of transferring that information over the network to the Subscriber. This improves synchronization performance over a slow network connection, but requires additional storage at the Publisher.
Attachable Subscription Databases	Snapshot Replication Transactional Replication Merge Replication	Attachable subscription databases allow you to transfer a database with replicated data and subscriptions from one Subscriber to another. After the database is attached to the new Subscriber, the database at the new Subscriber will automatically receive its own pull subscriptions to the publications at those Publishers. This saves you the time and effort of creating subscription databases and subscriptions at multiple Subscribers.

# How to validate Subscriber information (Transact-SQL)

When creating a merge publication, execute the sp\_addmergepublication system stored procedure and at the @validate\_subscriber\_info parameter, list the functions that are being used to retrieve Subscriber information.

For example, if you are using **SUSER\_SNAME()** in your dynamic filter, the parameter should read: **@validate\_subscriber\_info=N'SUSER\_SNAME()'**.

# Administering and Monitoring Replication (Transact-SQL)

SQL Server replication provides tools to administer and monitor replication agents, replication alerts, and replication processes so that you can ensure that replication is meeting the needs of your applications and your organization.

Monitoring replication will help you:

- Set the profiles, schedules and notifications for replication agents.
- Troubleshoot agent activity including verifying when agents last ran, monitoring agent activity.
- Troubleshoot agent errors.

Ensure that data values are the same at the Publisher and at Subscribers.

# How to create a replication agent profile (Transact-SQL) To create a replication agent profile.

- 1. Execute **sp\_add\_agent\_profile**.
- 2. Execute **sp\_add\_agent\_parameter**.

To set the default profile for a type of replication agent

- 1. Execute **sp\_update\_agent\_profile**.
- 2. Execute **sp\_help\_agent\_profile**.
- 3. Execute **sp\_help\_agent\_parameter**.

#### To view or modify a replication agent profile

- 1. Execute **sp\_help\_agent\_profile**.
- 2. Execute **sp\_help\_agent\_parameter**.
- 3. Execute **sp\_update\_agent\_profile**.

#### To delete a replication agent profile

• Execute **sp\_drop\_agent\_profile**.

## **Replication Security (Transact-SQL)**

Replication security is an important part of the design and implementation of your distributed application. Replication applies the data changes made elsewhere on the network to the database at your server and vice-versa.

The decentralized availability of replicated data increases the complexity of managing or restricting access to that data. Microsoft® SQL Server<sup>™</sup> 2000 replication uses a combination of security mechanisms to protect the data and business logic in your application:

• Role requirements

By mapping user logins to specific SQL Server 2000 roles, SQL Server 2000 allows users to perform only those replication and database activities authorized for that role. Replication grants certain permission to the **sysadmin** fixed server role, the **db\_owner** fixed database role, the current login, and the **public** role. For example, only members of the **sysadmin** server role can configure replication.

• Distributor administrative link security

SQL Server 2000 provides a secure administrative link between the Distributor and a remote Publisher. Publishers can be treated as trusted or nontrusted.

• Snapshot folder security

The operating system or FTP service prevents users from accessing specific files on the server. The user must have a valid login to read or write the files used in the replication process.

• Registered subscribers

SQL Server 2000 allows you to limit access to publications to either registered Subscribers that are well-known to the Publisher, anonymous, or Subscribers that have logins in the publication access list. SQL Server 2000 uses linked server definitions for heterogeneous Subscribers to secure the replication of data with heterogeneous data sources. • Publication access lists

By supporting publication access lists (PAL) on each server, SQL Server 2000 allows you to determine which logins have access to publications. SQL Server 2000 creates the PAL with default logins, but you can add or delete logins from the list.

• Agent login security

By supporting agent login security, SQL Server 2000 requires each user to supply a valid login account to connect to the server. Replication agents are required to use valid logins when connecting to Publishers, Distributors, and Subscribers. However, agents also can use different logins and security modes when connecting to different servers simultaneously.

• Immediate-updating Subscriber security

For immediate-updating Subscribers, SQL Server 2000 replication applies security mechanisms to the Publisher-RPC link and Publisher stored procedures.

When used together, these security mechanisms provide the highest safeguards for the data and business logic in your application.

## How to add or change a password on a Distributor

#### To add a password to a Distributor

• Execute **sp\_add\_distributor** specifying the *password* parameter.

#### To change a password on a Distributor

• Execute **sp\_changedistributor\_password**.

## To grant or revoke access to a publication

#### To grant access to a publication

• Execute **sp\_grant\_publication\_access**.

### To revoke access to a publication

• Execute **sp\_revoke\_publication\_access**.

## **OLE DB**

To use the Microsoft OLE DB Provider for SQL Server (SQLOLEDB), you have to understand how to make a connection to the server, execute the command, and process the results.

## **Processing Results (OLE DB)**

Processing results in an OLE DB application involves first determining the characteristics of the result set, and then retrieving the data into program variables. If the command executes a stored procedure, you also must know how to process return codes and output parameters from the stored procedures.

## Execute stored procedure (using ODBC CALL syntax) and process return codes and output parameters (OLE DB)

Microsoft® SQL Server<sup>™</sup> stored procedures can have integer return codes and output parameters. The return codes and output parameters are sent in the last packet from the server and are therefore not available to the application until the rowset is completely released. If the command returns multiple results, output parameter data is available when **IMultipleResults::GetResult** returns **DB\_S\_NORESULT** or the **IMultipleResults** interface is completely released, whichever occurs first.

#### To process return codes and output parameters

- 1. Construct an SQL statement that uses the ODBC CALL escape sequence. The statement should use parameter markers for each input/output, and output parameter, and for the procedure return value (if any). For input parameters, you can use the parameter markers or hard code the values.
- 2. Create a set of bindings (one for each parameter maker) by using an array of DBBINDING structure.
- 3. Create an accessor for the defined parameters by using the **IAccessor::CreateAccessor** method. **CreateAccessor** creates an accessor from a set of bindings.
- 4. Fill in the DBPARAMS structure.
- 5. Call the **Execute** command (in this case, a call to a stored procedure).
- 6. Process the rowset and release it by using the **IRowset::Release** method.

7. Process the return code and output parameter values received from the stored procedure.

This example shows processing a rowset, a return code, and an output parameter. Result sets are not processed. Here is the sample stored procedure used by the application.

USE pubs DROP PROCEDURE myProc GO

CREATE PROCEDURE myProc @inparam int, @outparam int OUTPUT

AS SELECT title, price FROM titles WHERE royalty > @inparam SELECT @outparam = 100

```
IF (@outparam > 0)
RETURN 999
ELSE
RETURN 888
GO
```

Here is the sample code:

void InitializeAndEstablishConnection();

#define UNICODE
#define DBINITCONSTANTS
#define INITGUID
#include <windows.h>

```
#include <stdio.h>
#include <stddef.h>
#include <iostream.h>
#include <oledb.h>
#include <oledberr.h>
#include <SQLOLEDB.h>
               pIDBInitialize
                                 = NULL;
IDBInitialize*
IDBCreateSession* pIDBCreateSession
                                       = NULL;
IDBCreateCommand* pIDBCreateCommand
                                            = NULL;
                  pICommandText
ICommandText*
                                       = NULL;
              pIRowset
                               = NULL;
IRowset*
ICommandWithParameters* pICommandWithParams = NULL;
                                = NULL;
               pIAccessor
IAccessor*
                pIDBProperties
IDBProperties*
                                   = NULL;
                pStringsBuffer;
WCHAR*
                  pBindings;
DBBINDING*
const ULONG
                 nInitProps = 4;
               InitProperties[nInitProps];
DBPROP
                 nPropSet = 1;
const ULONG
                  rgInitPropSet[nPropSet];
DBPROPSET
HRESULT
                hr:
HACCESSOR
                  hAccessor;
                 nParams = 3; //Number of parameters in the comm
const ULONG
DBPARAMBINDINFO
                       ParamBindInfo[nParams];
ULONG
               i;
               cbColOffset = 0;
ULONG
               ParamOrdinals[nParams];
ULONG
              cNumRows = 0;
LONG
DBPARAMS
                  Params;
/*
```

Declare an array of DBBINDING structures, one for each parameter

in the command.
\*/
DBBINDING acDBBinding[nParams];
DBBINDSTATUS acDBBindStatus[nParams];

//The following buffer is used to store parameter values.
typedef struct tagSPROCPARAMS
{
 long lReturnValue;

long outParam; long inParam; } SPROCPARAMS;

```
void main() {
```

```
//The command to execute.
WCHAR* wCmdString = L"{? = call myProc(?,?)}";
```

```
SPROCPARAMS sprocparams = {0,0,14};
```

//All the initialization activities in a separate function.
InitializeAndEstablishConnection();

```
cout << "Failed to access IDBCreateSession interface.\n";
goto EXIT;
```

```
IID IDBCreateCommand,
                   (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
  goto EXIT;
}
//Create a Command object.
if(FAILED(pIDBCreateCommand->CreateCommand(
                     NULL,
                     IID_ICommandText,
                     (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
  goto EXIT;
}
//Set the command text.
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
  cout << "Failed to set command text.\n";</pre>
  goto EXIT;
}
/* No need to describe command parameters (parameter name, data t
etc) in DBPARAMBINDINFO structure and then SetParameterInfo()
provider obtains this information by calling appropriate helper
function.
*/
```

/\*Describe the consumer buffer by filling in the array of DBBINDING structures. Each binding associates a single parameter to the consumer's buffer.\*/

```
for(i = 0; i < nParams; i++)
{
    acDBBinding[i].obLength = 0;
    acDBBinding[i].obStatus = 0;
    acDBBinding[i].pTypeInfo = NULL;
    acDBBinding[i].pObject = NULL;
    acDBBinding[i].pBindExt = NULL;
    acDBBinding[i].dwPart = DBPART_VALUE;
    acDBBinding[i].dwMemOwner = DBMEMOWNER_CLIENTOV
    acDBBinding[i].dwFlags = 0;
    acDBBinding[i].bScale = 0;
} //endfor</pre>
```

```
acDBBinding[0].iOrdinal = 1;
acDBBinding[0].obValue = offsetof(SPROCPARAMS, lReturnValu-
acDBBinding[0].eParamIO = DBPARAMIO_OUTPUT;
acDBBinding[0].cbMaxLen = sizeof(long);
acDBBinding[0].wType = DBTYPE_I4;
acDBBinding[0].bPrecision = 11;
```

```
acDBBinding[1].iOrdinal = 2;
acDBBinding[1].obValue = offsetof(SPROCPARAMS, inParam);
acDBBinding[1].eParamIO = DBPARAMIO_INPUT;
acDBBinding[1].cbMaxLen = sizeof(long);
acDBBinding[1].wType = DBTYPE_I4;
acDBBinding[1].bPrecision = 11;
```

```
acDBBinding[2].iOrdinal = 3;
acDBBinding[2].obValue = offsetof(SPROCPARAMS, outParam);
acDBBinding[2].eParamIO = DBPARAMIO_OUTPUT;
acDBBinding[2].cbMaxLen = sizeof(long);
acDBBinding[2].wType = DBTYPE_I4;
acDBBinding[2].bPrecision = 11;
```

```
//Create an accessor from the above set of bindings.
hr = pICommandText->QueryInterface(
                  IID_IAccessor,
                  (void**)&pIAccessor);
if (FAILED(hr))
{
  cout << "Failed to get IAccessor interface.\n";
}
hr = pIAccessor->CreateAccessor(
              DBACCESSOR PARAMETERDATA,
              nParams.
              acDBBinding,
              sizeof(SPROCPARAMS),
              &hAccessor,
              acDBBindStatus);
if (FAILED(hr))
{
 cout << "Failed to create accessor for the defined parameters.\n";
}
/*
Fill in DBPARAMS structure for the command execution. This struc
specifies the parameter values in the command and is then passed
to Execute.
*/
Params.pData = & sprocparams;
Params.cParamSets = 1;
Params.hAccessor = hAccessor;
//Execute the command.
if(FAILED(hr = pICommandText->Execute(
```

```
NULL,
```

```
IID_IRowset,
&Params,
&cNumRows,
(IUnknown **) &pIRowset)))
{
cout << "Failed to execute command.\n";
goto EXIT;
}
```

```
printf("After command execution but before rowset processing.\n\n"
printf(" Return value = %d\n", sprocparams.lReturnValue);
printf(" Output parameter value = %d\n", sprocparams.outParam);
printf(" These are the same default values set in the application.\n\n
```

```
/*The result set does not matter in this example, so release it
without processing.*/
pIRowset->Release();
```

```
printf("After processing the result set...\n");
printf(" Return value = %d\n", sprocparams.lReturnValue);
printf(" Output parameter value = %d\n\n", sprocparams.outParam)
```

```
//Free up memory.
pIAccessor->ReleaseAccessor(hAccessor, NULL);
pIAccessor->Release();
pICommandText->Release();
pIDBCreateCommand->Release();
pIDBCreateSession->Release();
if(FAILED(pIDBInitialize->Uninitialize()))
{
    /*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.*/
```

```
cout << "Problem uninitializing.\n";</pre>
```

```
} //endif
  pIDBInitialize->Release();
  //Release COM.
  CoUninitialize();
  return;
EXIT:
  if(pIAccessor != NULL)
     pIAccessor->Release();
  if(pICommandText != NULL)
     pICommandText->Release();
  if(pIDBCreateCommand != NULL)
    pIDBCreateCommand->Release();
  if(pIDBCreateSession != NULL)
    pIDBCreateSession->Release();
  if (pIDBInitialize != NULL)
  {
    if (FAILED(pIDBInitialize->Uninitialize()))
     {
       /*Uninitialize is not required, but it fails if an
       interface has not been released. This can be used
       for debugging.*/
       cout << "Problem in uninitializing.\n";</pre>
     } //if inner
    pIDBInitialize->Release();
  } //endif outer
  CoUninitialize();
};
//_____
void InitializeAndEstablishConnection()
{
```

```
//Initialize the COM library.
CoInitialize(NULL);
```

```
//Obtain access to the SQLOLEDB provider.
hr = CoCreateInstance(
         CLSID_SQLOLEDB,
         NULL,
         CLSCTX_INPROC_SERVER,
         IID IDBInitialize,
         (void **) &pIDBInitialize);
if (FAILED(hr))
{
  cout << "Failed in CoCreateInstance().\n";</pre>
}
/*
Initialize the property values needed
to establish the connection.
*/
for(i = 0; i < nInitProps; i++)</pre>
  VariantInit(&InitProperties[i].vValue);
//Specify server name.
InitProperties[0].dwPropertyID = DBPROP_INIT_DATASOURCE;
InitProperties[0].vValue.vt = VT BSTR;
InitProperties[0].vValue.bstrVal =
                SysAllocString(L"mohanv1");
InitProperties[0].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[0].colid = DB_NULLID;
//Specify database name.
InitProperties[1].dwPropertyID = DBPROP_INIT_CATALOG;
InitProperties[1].vValue.vt = VT_BSTR;
```

```
InitProperties[1].vValue.bstrVal =
```

```
SysAllocString(L"pubs");
InitProperties[1].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[1].colid = DB_NULLID;
```

//Specify password. InitProperties[3].dwPropertyID = DBPROP\_AUTH\_PASSWORD; InitProperties[3].vValue.vt = VT\_BSTR; InitProperties[3].vValue.bstrVal = SysAllocString(L'''); InitProperties[3].dwOptions = DBPROPOPTIONS\_REQUIRED; InitProperties[3].colid = DB\_NULLID;

/\*

Now that properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0] rgProperties = InitProperties;
```

rgInitPropSet[0].rgProperties = InitProperties;

```
{
     cout << "Failed to obtain IDBProperties interface.\n";</pre>
  }
  hr = pIDBProperties->SetProperties(
                     nPropSet,
                     rgInitPropSet);
  if(FAILED(hr))
  {
     cout << "Failed to set initialization properties.\n";</pre>
  }
  pIDBProperties->Release();
  //Now establish a connection to the data source.
  if(FAILED(pIDBInitialize->Initialize()))
  {
     cout << "Problem in initializing.\n";</pre>
  }
} //end of InitializeAndEstablishConnection.
```

# Execute stored procedure (using RPC syntax) and process return codes and output parameters (OLE DB)

Microsoft® SQL Server<sup>™</sup> stored procedures can have integer return codes and output parameters. The return codes and output parameters are sent in the last packet from the server and are therefore not available to the application until the rowset is completely released. If the command returns multiple results, output parameter data is available when **IMultipleResults::GetResult** returns **DB\_S\_NORESULT** or the **IMultipleResults** interface is completely released, whichever occurs first.

#### To process return codes and output parameters

- 1. Construct an SQL statement that uses the RPC escape sequence.
- 2. Call the **ICommandWithParameters::SetParameterInfo** method to describe parameters to the provider. Fill in the parameter information in an array of PARAMBINDINFO structures.
- 3. Create a set of bindings (one for each parameter maker) by using an array of DBBINDING structure.
- 4. Create an accessor for the defined parameters by using the **IAccessor::CreateAccessor** method. **CreateAccessor** creates an accessor from a set of bindings.
- 5. Fill in the DBPARAMS structure.
- 6. Call the **Execute** command (in this case, a call to a stored procedure).
- 7. Process the rowset and release it by using the **IRowset::Release**

method.

8. Process the return code and output parameter values received from the stored procedure.

This example shows processing a rowset, a return code, and an output parameter. Result sets are not processed. Here is the sample stored procedure used by the application.

USE pubs DROP PROCEDURE myProc GO

CREATE PROCEDURE myProc @inparam int, @outparam int OUTPUT

AS SELECT title, price FROM titles WHERE royalty > @inparam SELECT @outparam = 100

```
IF (@outparam > 0)
RETURN 999
ELSE
RETURN 888
GO
```

Here is the sample code: void InitializeAndEstablishConnection();

#define UNICODE #define DBINITCONSTANTS #define INITGUID

```
#include <windows.h>
#include <stdio.h>
#include <stddef.h>
#include <iostream.h>
#include <oledb.h>
#include <oledberr.h>
#include <SQLOLEDB.h>
IDBInitialize*
               pIDBInitialize
                                 = NULL:
IDBCreateSession* pIDBCreateSession
                                       = NULL;
IDBCreateCommand* pIDBCreateCommand
                                            = NULL;
ICommandText*
                  pICommandText
                                       = NULL;
                               = NULL;
IRowset*
              pIRowset
ICommandWithParameters* pICommandWithParams = NULL;
                                = NULL;
IAccessor*
               pIAccessor
                pIDBProperties
                                   = NULL;
IDBProperties*
WCHAR*
                pStringsBuffer;
                  pBindings;
DBBINDING*
                 nInitProps = 4;
const ULONG
               InitProperties[nInitProps];
DBPROP
                 nPropSet = 1;
const ULONG
                  rgInitPropSet[nPropSet];
DBPROPSET
HRESULT
                hr:
HACCESSOR
                  hAccessor:
                 nParams = 3; //Number of parameters in the comm
const ULONG
DBPARAMBINDINFO
                       ParamBindInfo[nParams];
ULONG
               i;
ULONG
               cbColOffset = 0;
               ParamOrdinals[nParams];
ULONG
LONG
              cNumRows = 0;
DBPARAMS
                  Params;
/*
```

Declare an array of DBBINDING structures, one for each parameter in the command.

\*/

```
DBBINDING acDBBinding[nParams];
DBBINDSTATUS acDBBindStatus[nParams];
```

```
//The following buffer is used to store parameter values. typedef struct tagSPROCPARAMS
```

```
{
    long lReturnValue;
    long outParam;
    long inParam;
} CDDOCDADAMC;
```

```
} SPROCPARAMS;
```

```
void main() {
```

```
//The command to execute.
//WCHAR* wCmdString = L"{? = call myProc(?,?)}";
WCHAR* wCmdString=L"{rpc myProc}";
SPROCPARAMS sprocparams = {0,0,14};
```

//All the initialization activities in a separate function.
InitializeAndEstablishConnection();

## {

```
cout << "Failed to access IDBCreateSession interface.\n";
goto EXIT;
}</pre>
```

```
if(FAILED(pIDBCreateSession->CreateSession(
```

```
NULL,
                  IID_IDBCreateCommand,
                  (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
  goto EXIT;
}
//Create a Command object.
if(FAILED(pIDBCreateCommand->CreateCommand(
                    NULL,
                    IID ICommandText,
                    (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
  goto EXIT;
}
//Set the command text.
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
  cout << "Failed to set command text.\n";
  goto EXIT;
}
/*
Describe the command parameters (parameter name, provider
specific name of the parameter's data type, and so on.) in an array of
```

specific name of the parameter's data type, and so on.) in an array of DBPARAMBINDINFO structures. This information is then used by SetParameterInfo().

\*/

```
ParamBindInfo[0].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[0].pwszName = L"ReturnVal"; //return value from s
```

```
ParamBindInfo[0].ulParamSize = sizeof(long);
ParamBindInfo[0].dwFlags = DBPARAMFLAGS_ISOUTPUT;
ParamBindInfo[0].bPrecision = 11;
ParamBindInfo[0].bScale = 0;
ParamOrdinals[0] = 1;
```

```
ParamBindInfo[1].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[1].pwszName = L"@inparam";
ParamBindInfo[1].ulParamSize = sizeof(long);
ParamBindInfo[1].dwFlags = DBPARAMFLAGS_ISINPUT;
ParamBindInfo[1].bPrecision = 11;
ParamBindInfo[1].bScale = 0;
ParamOrdinals[1] = 2;
```

```
ParamBindInfo[2].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[2].pwszName = L"@outparam";
ParamBindInfo[2].ulParamSize = sizeof(long);
ParamBindInfo[2].dwFlags = DBPARAMFLAGS_ISOUTPUT;
ParamBindInfo[2].bPrecision = 11;
ParamBindInfo[2].bScale = 0;
ParamOrdinals[2] = 3;
```

```
{
    cout << "Failed in setting parameter information.(SetParameterIn:
    goto EXIT;
}</pre>
```

```
/*Describe the consumer buffer by filling in the array
of DBBINDING structures. Each binding associates
a single parameter to the consumer's buffer.*/
for(i = 0; i < nParams; i++)
{
    acDBBinding[i].obLength = 0;
    acDBBinding[i].obStatus = 0;
    acDBBinding[i].pTypeInfo = NULL;
    acDBBinding[i].pObject = NULL;
    acDBBinding[i].pBindExt = NULL;
    acDBBinding[i].dwPart = DBPART_VALUE;
    acDBBinding[i].dwMemOwner = DBMEMOWNER_CLIENTOV
    acDBBinding[i].dwFlags = 0;
    acDBBinding[i].bScale = 0;
} //endfor</pre>
```

```
acDBBinding[0].iOrdinal = 1;
acDBBinding[0].obValue = offsetof(SPROCPARAMS, lReturnValu-
acDBBinding[0].eParamIO = DBPARAMIO_OUTPUT;
acDBBinding[0].cbMaxLen = sizeof(long);
acDBBinding[0].wType = DBTYPE_I4;
acDBBinding[0].bPrecision = 11;
```

```
acDBBinding[1].iOrdinal = 2;
acDBBinding[1].obValue = offsetof(SPROCPARAMS, inParam);
acDBBinding[1].eParamIO = DBPARAMIO_INPUT;
acDBBinding[1].cbMaxLen = sizeof(long);
```

```
acDBBinding[1].wType = DBTYPE_I4;
acDBBinding[1].bPrecision = 11;
```

```
acDBBinding[2].iOrdinal = 3;
acDBBinding[2].obValue = offsetof(SPROCPARAMS, outParam);
acDBBinding[2].eParamIO = DBPARAMIO_OUTPUT;
acDBBinding[2].cbMaxLen = sizeof(long);
acDBBinding[2].wType = DBTYPE_I4;
acDBBinding[2].bPrecision = 11;
```

```
//Create an accessor from the above set of bindings.
hr = pICommandWithParams->QueryInterface(
```

```
IID_IAccessor,
(void**)&pIAccessor);
```

```
if (FAILED(hr))
```

```
{
```

```
cout << "Failed to get IAccessor interface.\n";</pre>
```

```
}
```

```
hr = pIAccessor->CreateAccessor(
DBACCESSOR PARAMETERDATA,
```

```
nParams,
acDBBinding,
sizeof(SPROCPARAMS),
&hAccessor,
```

```
acDBBindStatus);
```

```
if (FAILED(hr))
```

```
{
```

```
cout << "Failed to create accessor for the defined parameters.\n";
}</pre>
```

/\*

Fill in DBPARAMS structure for the command execution. This struc specifies the parameter values in the command and is then passed

```
to Execute.
*/
Params.pData = & sprocparams;
Params.cParamSets = 1;
Params.hAccessor = hAccessor:
//Execute the command.
if(FAILED(hr = pICommandText->Execute(
                  NULL,
                  IID_IRowset,
                  &Params,
                  &cNumRows,
                  (IUnknown **) &pIRowset)))
{
  cout << "Failed to execute command.\n";</pre>
  goto EXIT;
}
```

```
printf("After command execution but before rowset processing.\n\n"
printf(" Return value = %d\n", sprocparams.lReturnValue);
printf(" Output parameter value = %d\n", sprocparams.outParam);
printf(" These are the same default values set in the application.\n\n
```

```
/*The result set does not matter in this example, so release it
without processing.*/
pIRowset->Release();
```

```
printf("After processing the result set...\n");
printf(" Return value = %d\n", sprocparams.lReturnValue);
printf(" Output parameter value = %d\n\n", sprocparams.outParam);
```

```
//Free up memory.
pIAccessor->ReleaseAccessor(hAccessor, NULL);
```

```
pIAccessor->Release();
  pICommandWithParams->Release();
  pICommandText->Release();
  pIDBCreateCommand->Release();
  pIDBCreateSession->Release();
  if(FAILED(pIDBInitialize->Uninitialize()))
  ł
    /*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.*/
    cout << "Problem uninitializing.\n";</pre>
  } //endif
  pIDBInitialize->Release();
  //Release COM.
  CoUninitialize();
  return;
EXIT:
  if(pIAccessor != NULL)
    pIAccessor->Release();
  if(pICommandWithParams != NULL)
    pICommandWithParams->Release();
  if(pICommandText != NULL)
    pICommandText->Release();
  if(pIDBCreateCommand != NULL)
    pIDBCreateCommand->Release();
  if(pIDBCreateSession != NULL)
   pIDBCreateSession->Release();
  if (pIDBInitialize != NULL)
  ł
    if (FAILED(pIDBInitialize->Uninitialize()))
    {
```

/\*Uninitialize is not required, but it fails if an

```
interface has not been released. This can be used
       for debugging.*/
       cout << "Problem in uninitializing.\n";</pre>
    } //if inner
    pIDBInitialize->Release();
  } //endif outer
  CoUninitialize();
};
//_____
void InitializeAndEstablishConnection()
{
  //Initialize the COM library.
  CoInitialize(NULL);
  //Obtain access to the SQLOLEDB provider.
  hr = CoCreateInstance(
           CLSID_SQLOLEDB,
           NULL,
           CLSCTX INPROC SERVER,
           IID IDBInitialize,
           (void **) &pIDBInitialize);
  if (FAILED(hr))
  {
    cout << "Failed in CoCreateInstance().\n";</pre>
  }
  /*
  Initialize the property values needed
  to establish the connection.
  */
  for(i = 0; i < nInitProps; i++)</pre>
    VariantInit(&InitProperties[i].vValue);
```

```
//Specify password.
InitProperties[3].dwPropertyID = DBPROP_AUTH_PASSWORD;
InitProperties[3].vValue.vt = VT_BSTR;
InitProperties[3].vValue.bstrVal = SysAllocString(L''');
InitProperties[3].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[3].colid = DB_NULLID;
```

```
/*
```

Now that properties are set, construct the DBPROPSET structure

```
(rgInitPropSet). The DBPROPSET structure is used to pass an array
of DBPROP structures (InitProperties) to the SetProperties method.
*/
```

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0].rgProperties = InitProperties;
```

```
//Set initialization properties.
hr = pIDBInitialize->QueryInterface(
                 IID_IDBProperties,
                 (void **)&pIDBProperties);
if (FAILED(hr))
{
  cout << "Failed to obtain IDBProperties interface.\n";
}
hr = pIDBProperties->SetProperties(
                 nPropSet,
                 rgInitPropSet);
if(FAILED(hr))
{
  cout << "Failed to set initialization properties.\n";
}
pIDBProperties->Release();
//Now establish a connection to the data source.
if(FAILED(pIDBInitialize->Initialize()))
ł
  cout << "Problem in initializing.\n";</pre>
```

```
} //end of InitializeAndEstablishConnection.
```

How To

# **Execute user-defined function and process return code** (OLE DB)

In this example a user-defined function is executed and the return code is printed. Here is the sample user-defined function used by the application.

```
drop function fn_RectangleArea
go
CREATE FUNCTION fn_RectangleArea
(@Width int,
@Height int )
RETURNS int
AS
BEGIN
RETURN (@Width * @Height )
END
GO
```

Here is a sample code.

/\*

Example shows how to execute user-defined functions and process retu $\ast/$ 

```
void InitializeAndEstablishConnection();
#define UNICODE
#define DBINITCONSTANTS
#define INITGUID
#include <windows.h>
#include <stdio.h>
#include <stddef.h>
#include <iostream.h>
#include <oledb.h>
#include <oledb.h>
#include <oledb.h>
```

#include <sqloledb.h>

```
IDBInitialize*
               pIDBInitialize
                                 = NULL:
IDBCreateSession* pIDBCreateSession
                                       = NULL;
IDBCreateCommand* pIDBCreateCommand
                                           = NULL:
ICommandText*
                  pICommandText
                                      = NULL;
              pIRowset
                               = NULL:
IRowset*
ICommandWithParameters* pICommandWithParams = NULL;
                                = NULL:
IAccessor*
               pIAccessor
IDBProperties*
                pIDBProperties
                                   = NULL;
WCHAR*
                pStringsBuffer;
                  pBindings;
DBBINDING*
                 nInitProps = 4;
const ULONG
               InitProperties[nInitProps];
DBPROP
                 nPropSet = 1;
const ULONG
                 rgInitPropSet[nPropSet];
DBPROPSET
HRESULT
                hr:
HACCESSOR
                  hAccessor:
const ULONG
                 nParams = 3; //No. of parameters in the command
                       ParamBindInfo[nParams];
DBPARAMBINDINFO
ULONG
               i:
               cbColOffset = 0;
ULONG
               ParamOrdinals[nParams];
ULONG
              cNumRows = 0;
LONG
DBPARAMS
                  Params;
/*
Declare an array of DBBINDING structures, one for each parameter
in the command
*/
DBBINDING
                 acDBBinding[nParams];
                    acDBBindStatus[nParams];
DBBINDSTATUS
```

```
//The following buffer is used to store parameter values.
typedef struct tagSPROCPARAMS
{
    long lReturnValue;
    long inParam1;
    long inParam2;
} SPROCPARAMS;
```

```
void main() {
```

//The command to execute.

```
WCHAR* wCmdString = L"{? = CALL fn_RectangleArea(?, ?) }";
// WCHAR* wCmdString = L"EXEC ? = fn_RectangleVolume(?, ?)";
```

```
SPROCPARAMS sprocparams = {0,5,10};
```

```
//All the initialization stuff in a separate function.
InitializeAndEstablishConnection();
```

```
{
```

```
cout << "Failed to access IDBCreateSession interface\\n";
goto EXIT;
```

```
}
```

```
if(FAILED(pIDBCreateSession->CreateSession(
```

```
NULL,
```

IID\_IDBCreateCommand,

```
(IUnknown**) &pIDBCreateCommand)))
```

```
{
```

```
cout << "pIDBCreateSession->CreateSession failed\\n";
```

```
goto EXIT;
}
//Create a Command
```

```
//Set the command text.
```

```
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
```

```
cout << "failed to set command text\\n";
goto EXIT;
```

```
}
```

```
/*
```

Describe the command parameters (parameter name, provider specific name of the parameter's data type etc.) in an array of DBPARAMBINDINFO structures. This information is then used by SetParameterInfo().

```
*/
```

```
ParamBindInfo[0].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[0].pwszName = NULL; // L"ReturnVal"; //return va
ParamBindInfo[0].ulParamSize = sizeof(long);
ParamBindInfo[0].dwFlags = DBPARAMFLAGS_ISOUTPUT;
ParamBindInfo[0].bPrecision = 11;
ParamBindInfo[0].bScale = 0;
ParamOrdinals[0] = 1;
```

```
ParamBindInfo[1].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[1].pwszName = NULL; //L"@inparam1";
ParamBindInfo[1].ulParamSize = sizeof(long);
ParamBindInfo[1].dwFlags = DBPARAMFLAGS_ISINPUT;
ParamBindInfo[1].bPrecision = 11;
ParamBindInfo[1].bScale = 0;
ParamOrdinals[1] = 2;
```

```
ParamBindInfo[2].pwszDataSourceType = L"DBTYPE_I4";
ParamBindInfo[2].pwszName = NULL; // L"@inparam2";
ParamBindInfo[2].ulParamSize = sizeof(long);
ParamBindInfo[2].dwFlags = DBPARAMFLAGS_ISINPUT;
ParamBindInfo[2].bPrecision = 11;
ParamBindInfo[2].bScale = 0;
ParamOrdinals[2] = 3;
```

```
}
```

goto EXIT;

```
//Let us describe the consumer buffer by filling in the array
//of DBBINDING structures. Each binding associates
//a single parameter to the consumer's buffer.
for(i = 0; i < nParams; i++)
{
    acDBBinding[i].obLength = 0;
    acDBBinding[i].obStatus = 0;
    acDBBinding[i].pTypeInfo = NULL;
    acDBBinding[i].pObject = NULL;
    acDBBinding[i].pBindExt = NULL;
    acDBBinding[i].pBindExt = NULL;
    acDBBinding[i].dwPart = DBPART_VALUE;
    acDBBinding[i].dwFlags = 0;
    acDBBinding[i].bScale = 0;
} //for</pre>
```

```
acDBBinding[0].iOrdinal = 1;
acDBBinding[0].obValue = offsetof(SPROCPARAMS, lReturnValu-
acDBBinding[0].eParamIO = DBPARAMIO_OUTPUT;
acDBBinding[0].cbMaxLen = sizeof(long);
acDBBinding[0].wType = DBTYPE_I4;
acDBBinding[0].bPrecision = 11;
```

```
acDBBinding[1].iOrdinal = 2;
acDBBinding[1].obValue = offsetof(SPROCPARAMS, inParam1);
acDBBinding[1].eParamIO = DBPARAMIO_INPUT;
acDBBinding[1].cbMaxLen = sizeof(long);
acDBBinding[1].wType = DBTYPE_I4;
acDBBinding[1].bPrecision = 11;
```

```
acDBBinding[2].iOrdinal = 3;
acDBBinding[2].obValue = offsetof(SPROCPARAMS, inParam2);
```

```
acDBBinding[2].eParamIO = DBPARAMIO_INPUT;
acDBBinding[2].cbMaxLen = sizeof(long);
acDBBinding[2].wType = DBTYPE_I4;
acDBBinding[2].bPrecision = 11;
```

```
//Let us create an accessor from the above set of bindings.
hr = pICommandWithParams->QueryInterface(
                  IID IAccessor,
                  (void**)&pIAccessor);
if (FAILED(hr))
{
  cout << "Failed to get IAccessor interface\\n";
}
hr = pIAccessor->CreateAccessor(
              DBACCESSOR PARAMETERDATA,
              nParams,
              acDBBinding,
              sizeof(SPROCPARAMS),
              &hAccessor,
              acDBBindStatus);
if (FAILED(hr))
{
  cout << "failed to create accessor for the defined parameters\\n";
}
/*
Fill in DBPARAMS structure for the command execution. This struc
specify the parameter values in the command. This structure is
then passed to Execute.
*/
Params.pData = & sprocparams;
Params.cParamSets = 1;
```

```
Params.hAccessor = hAccessor;
```

```
printf(" Return value = %d\n", sprocparams.lReturnValue);
```

```
//we are not interested in the resultset so release it
//without processing.
if (pIRowset != NULL)
 pIRowset->Release();
//Free up memory.
pIAccessor->ReleaseAccessor(hAccessor, NULL);
pIAccessor->Release();
pICommandWithParams->Release();
pICommandText->Release();
pIDBCreateCommand->Release();
pIDBCreateSession->Release();
if(FAILED(pIDBInitialize->Uninitialize()))
ł
   //Uninitialize is not required, but it fails if an interface
   //has not not been released. This can be used for debugging.
   cout << "Problem uninitializing\\n";</pre>
} //if
pIDBInitialize->Release();
```

```
//Release COM.
CoUninitialize();
return;
```

EXIT:

}

{

```
if(pIAccessor != NULL)
    pIAccessor->Release();
  if(pICommandWithParams != NULL)
    pICommandWithParams->Release();
  if(pICommandText != NULL)
    pICommandText->Release();
  if(pIDBCreateCommand != NULL)
    pIDBCreateCommand->Release();
  if(pIDBCreateSession != NULL)
    pIDBCreateSession->Release();
  if (pIDBInitialize != NULL)
  {
    if (FAILED(pIDBInitialize->Uninitialize()))
    {
       //Uninitialize is not required, but it fails if an
       //interface has not been released. This can be used
       //for debugging.
       cout << "problem in uninitializing\\n";
    } //if inner
    pIDBInitialize->Release();
  } //if outer
  CoUninitialize();
//_____
void InitializeAndEstablishConnection()
```

```
//Initialize the COM library.
CoInitialize(NULL);
```

```
//Obtain access to the SQLOLEDB provider.
hr = CoCreateInstance(
         CLSID_SQLOLEDB,
         NULL,
         CLSCTX_INPROC_SERVER,
         IID IDBInitialize,
         (void **) &pIDBInitialize);
if (FAILED(hr))
{
  cout << "Failed in CoCreateInstance()\\n";</pre>
}
/*
Let us initialize the property values needed
to establish the connection.
*/
for(i = 0; i < nInitProps; i++)</pre>
  VariantInit(&InitProperties[i].vValue);
//Specify server name.
InitProperties[0].dwPropertyID = DBPROP_INIT_DATASOURCE;
InitProperties[0].vValue.vt = VT_BSTR;
InitProperties[0].vValue.bstrVal =
                SysAllocString(L"server");
InitProperties[0].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[0].colid = DB_NULLID;
//Specify database name.
InitProperties[1].dwPropertyID = DBPROP INIT CATALOG;
InitProperties[1].vValue.vt = VT_BSTR;
```

```
InitProperties[1].vValue.bstrVal =
```

```
SysAllocString(L"database");
InitProperties[1].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[1].colid = DB_NULLID;
```

```
//Specify password.
```

```
InitProperties[3].dwPropertyID = DBPROP_AUTH_PASSWORD;
InitProperties[3].vValue.vt = VT_BSTR;
InitProperties[3].vValue.bstrVal = SysAllocString(L"password");
InitProperties[3].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[3].colid = DB_NULLID;
```

/\*

Now that properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0] rgProperties = InitProperties;
```

```
rgInitPropSet[0].rgProperties = InitProperties;
```

```
{
     cout << "Failed to obtain IDBProperties interface.\\n";</pre>
  }
  hr = pIDBProperties->SetProperties(
                     nPropSet,
                     rgInitPropSet);
  if(FAILED(hr))
  {
     cout << "Failed to set initialization properties\\n";</pre>
   }
  pIDBProperties->Release();
  //Now we establish connection to the data source.
  if(FAILED(pIDBInitialize->Initialize()))
   {
     cout << "Problem in initializing\\n";</pre>
  }
} //end of InitializeAndEstablishConnection.
```

How To

## How to fetch rows from a result set (OLE DB)

#### To fetch rows from a result set

/\*

Example shows how to fetch rows from a result set. \*/ void InitializeAndEstablishConnection(); void ProcessResultSet();

#define UNICODE
#define \_UNICODE
#define DBINITCONSTANTS
#define INITGUID

#include <stdio.h>
#include <tchar.h>
#include <stddef.h>
#include <windows.h>
#include <iostream.h>
#include <oledb.h>
#include <SQLOLEDB.h>

pIDBInitialize = NULL; IDBInitialize\* IDBProperties\* pIDBProperties = NULL; IDBCreateSession\* pIDBCreateSession = NULL; IDBCreateCommand\* pIDBCreateCommand = NULL; pICommandText ICommandText\* = NULL: IRowset\* pIRowset = NULL; pIColumnsInfo = NULL; IColumnsInfo\* pDBColumnInfo DBCOLUMNINFO\* = NULL; IAccessor\* pIAccessor = NULL;

DBPROP	InitProperties[4];
DBPROPSET	rgInitPropSet[1];
ULONG	i, j;
HRESULT	hr;
LONG	cNumRows = 0;
ULONG	lNumCols;
WCHAR*	pStringsBuffer;
DBBINDING*	pBindings;
ULONG	ConsumerBufColOffset = 0;
HACCESSOR	hAccessor;
ULONG	lNumRowsRetrieved;
HROW	hRows[10];
HROW*	pRows = &hRows[0];
BYTE*	pBuffer;

```
void main() {
```

//Here is the command to execute.

```
WCHAR* wCmdString
```

```
= OLESTR(" SELECT title, price FROM titles WHERE royalty >
// Call a function to initialize and establish connection.
InitializeAndEstablishConnection();
```

```
if(FAILED(pIDBCreateSession->CreateSession(
NULL,
```

```
IID_IDBCreateCommand,
                  (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
}
//Access the ICommandText interface.
if(FAILED(pIDBCreateCommand->CreateCommand(
                  NULL,
                  IID_ICommandText,
                  (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
}
//Use SetCommandText() to specify the command text.
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
  cout << "Failed to set command text.\n";</pre>
}
//Execute the command.
if(FAILED(hr = pICommandText->Execute(NULL,
                  IID_IRowset,
                  NULL,
                  &cNumRows,
                  (IUnknown **) &pIRowset)))
{
  cout << "Failed to execute command.\n";
}
//Process the result set.
ProcessResultSet();
```

```
pIRowset->Release();
```

```
//Free up memory.
pICommandText->Release();
pIDBCreateCommand->Release();
pIDBCreateSession->Release();
```

```
if(FAILED(pIDBInitialize->Uninitialize()))
  {
    /*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.
    cout << "Problem uninitializing.\n"; */</pre>
  } //endif.
  pIDBInitialize->Release();
  //Release the COM library.
  CoUninitialize();
};
//_____
void InitializeAndEstablishConnection()
{
  //Initialize the COM library.
  CoInitialize(NULL);
  //Obtain access to the SQLOLEDB provider.
  hr = CoCreateInstance(CLSID_SQLOLEDB,
               NULL,
               CLSCTX_INPROC_SERVER,
               IID IDBInitialize,
              (void **) &pIDBInitialize);
  if(FAILED(hr))
```

```
{
```

```
printf("Failed to get IDBInitialize interface.\n");
} //end if
```

#### /\*

```
Initialize the property values needed
to establish the connection.
*/
for(i = 0; i < 4; i++)
```

```
VariantInit(&InitProperties[i].vValue);
```

```
//Server name.
```

```
InitProperties[0].dwPropertyID = DBPROP_INIT_DATASOURCE
InitProperties[0].vValue.vt = VT_BSTR;
```

```
InitProperties[0].vValue.bstrVal=
```

```
SysAllocString(L"server");
InitProperties[0].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[0].colid = DB_NULLID;
```

//Database.

```
InitProperties[1].dwPropertyID = DBPROP_INIT_CATALOG;
InitProperties[1].vValue.vt = VT_BSTR;
InitProperties[1].vValue.bstrVal= SysAllocString(L"database");
InitProperties[1].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[1].colid = DB_NULLID;
```

```
//Username (login).
```

```
InitProperties[2].dwPropertyID = DBPROP_AUTH_USERID;
InitProperties[2].vValue.vt = VT_BSTR;
InitProperties[2].vValue.bstrVal= SysAllocString(L"login");
InitProperties[2].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[2].colid = DB_NULLID;
```

//Password.

```
InitProperties[3].dwPropertyID = DBPROP_AUTH_PASSWORD;
InitProperties[3].vValue.vt = VT_BSTR;
InitProperties[3].vValue.bstrVal= SysAllocString(L"Password");
InitProperties[3].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[3].colid = DB_NULLID;
```

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
```

```
rgInitPropSet[0].rgProperties = InitProperties;
```

```
//Set initialization properties.
```

```
hr = pIDBInitialize->QueryInterface(IID_IDBProperties,
```

```
(void **)&pIDBProperties);
```

```
if(FAILED(hr))
```

```
cout << "Failed to get IDBProperties interface.\n";
```

```
}
```

```
hr = pIDBProperties->SetProperties(1, rgInitPropSet);
if(FAILED(hr))
```

```
{
```

```
cout << "Failed to set initialization properties.\n";
} //end if</pre>
```

```
pIDBProperties->Release();
```

//Now establish the connection to the data source.

```
if(FAILED(pIDBInitialize->Initialize()))
  {
    cout << "Problem in establishing connection to the data
    source.\n";
  }
} //end of InitializeAndEstablishConnection.
//_____
                     _____
//Retrieve and display data resulting from a query.
void ProcessResultSet()
{
  //Obtain access to the IColumnInfo interface, from the Rowset
  object.
  hr = pIRowset->QueryInterface(IID_IColumnsInfo,
                   (void **)&pIColumnsInfo);
  if(FAILED(hr))
  ł
    cout << "Failed to get IColumnsInfo interface.\n";
  } //end if
  //Retrieve the column information.
  pIColumnsInfo->GetColumnInfo(&lNumCols,
                   &pDBColumnInfo,
                   &pStringsBuffer);
  //Free the column information interface.
  pIColumnsInfo->Release();
  //Create a DBBINDING array.
```

```
pBindings = new DBBINDING[lNumCols];
```

//Using the ColumnInfo structure, fill out the pBindings array.
for(j=0; j<lNumCols; j++) {
 pBindings[j].iOrdinal = j+1;</pre>

```
pBindings[j].obValue = ConsumerBufColOffset;
    pBindings[j].pTypeInfo = NULL;
    pBindings[j].pObject = NULL;
    pBindings[j].pBindExt = NULL;
    pBindings[j].dwPart = DBPART VALUE;
    pBindings[j].dwMemOwner = DBMEMOWNER_CLIENTOWN
    pBindings[j].eParamIO = DBPARAMIO NOTPARAM;
    pBindings[j].cbMaxLen = pDBColumnInfo[j].ulColumnSize;
    pBindings[j].dwFlags = 0;
    pBindings[j].wType = pDBColumnInfo[j].wType;
    pBindings[j].bPrecision = pDBColumnInfo[j].bPrecision;
    pBindings[j].bScale = pDBColumnInfo[j].bScale;
    //Compute the next buffer offset.
    ConsumerBufColOffset = ConsumerBufColOffset +
                 pDBColumnInfo[j].ulColumnSize;
  };
  //Get the IAccessor interface.
  hr = pIRowset->QueryInterface(IID_IAccessor, (void **) &pIAcces
  if(FAILED(hr))
  {
    cout << "Failed to obtain IAccessor interface.\n";
//Create an accessor from the set of bindings (pBindings).
  pIAccessor->CreateAccessor(DBACCESSOR_ROWDATA,
                  lNumCols.
                  pBindings,
                  0,
                  &hAccessor,
                  NULL);
  //Print column names.
  for(j=0; j<lNumCols; j++) {</pre>
```

}; //for.

```
//Release the rows retrieved.
pIRowset->ReleaseRows(lNumRowsRetrieved,
hRows,
NULL,
NULL,
NULL);
```

} //ProcessResultSet.

How To

# **Processing Large Data**

SQLOLEDB exposes the **ISequentialStream** interface to support consumer access to Microsoft® SQL Server<sup>™</sup> ntext, text and image data types as binary large objects (BLOBs).

How To

# How to set large data (OLE DB)

To pass a pointer to its own storage object, the consumer creates an accessor that binds the value of the BLOB column and then calls the

IRowsetChange::SetData or IRowsetChange::InsertRow methods.

#### To set BLOB data

- Create a DBOBJECT structure describing how the BLOB column should be accessed. Set the *dwFlag* element of the DBOBJECT structure to **STGM\_READ** and set the *iid* element to **IID\_ISequentialStream** (the interface to be exposed).
- 2. Set the properties in the **DBPROPSET\_ROWSET** property group so the rowset is updatable.
- 3. Create a set of bindings (one of each column) by using an array of DBBINDING structures. Set the *wType* element in the DBBINDING structure to **DBTYPE\_IUNKNOWN**, and the *pObject* element to point to the DBOBJECT structure you created.
- 4. Create an accessor using the binding information in the DBBINDINGS array of structures.
- 5. Call **GetNextRows** to fetch next rows into the rowset. Call **GetData** to read the data from the rowset.
- 6. To set the data, create a storage object containing the data (and also the length indicator), and then call IRowsetChange::SetData (or IRowsetChange::InsertRow) with the accessor that binds the BLOB column.

This example shows how to set BLOB data. The example creates a table, adds a sample record, fetches that record in the rowset, and then sets the value of the

BLOB field:

#define UNICODE #define DBINITCONSTANTS #define INITGID

```
#include <windows.h>
#include <stdio.h>
#include <stddef.h>
#include <iostream.h>
```

```
#include <oledb.h>
#include <oledberr.h>
```

```
#include <SQLOLEDB.h>
```

```
#define SAFE_RELEASE(pIUnknown) if(pIUnknown) (pIUnknown)-
HRESULT GetCommandObject(REFIID riid, IUnknown** ppIUnkno<sup>-</sup>
HRESULT CreateTable(ICommandText* pICommandText);
```

```
class CSeqStream : public ISequentialStream
{
public:
    //Constructors
    CSeqStream();
    virtual ~CSeqStream();

    virtual BOOL Seek(ULONG iPos);
    virtual BOOL Clear();
```

```
virtual BOOL CompareData(void* pBuffer);
```

```
virtual ULONG Length() { return m_cBufSize; };
```

```
virtual operator void* const() { return m_pBuffer; };
```

### STDMETHODIMP\_(ULONG) AddRef(void); STDMETHODIMP\_(ULONG) Release(void); STDMETHODIMP QueryInterface(REFIID riid, LPVOID \*ppv);

## STDMETHODIMP Read(

/\* [in] \*/ ULONG cb,

/\* [out] \*/ ULONG \_\_\_RPC\_FAR \*pcbRead);

## STDMETHODIMP Write(

/\* [in] \*/ const void \_\_\_RPC\_FAR \*pv,

/\* [in] \*/ ULONG cb,

/\* [out]\*/ ULONG \_\_\_\_RPC\_FAR \*pcbWritten);

protected:

//Data

private:

ULONG	m_cRef;	// reference count
void*	m_pBuffer;	// buffer
ULONG	m_cBufSi	ize; // buffer size
ULONG	f m_iPos;	<pre>// current index position in the buffer</pre>
};		

//class implementation

```
CSeqStream::CSeqStream()
{
    m_iPos = 0;
    m_cRef = 0;
    m_pBuffer = NULL;
    m_cBufSize = 0;
```

```
//The constructor AddRef's
  AddRef();
}
CSeqStream::~CSeqStream()
{
  //Shouldn't have any references left
// ASSERT(m_cRef == 0);
  CoTaskMemFree(m_pBuffer);
}
ULONG CSeqStream::AddRef(void)
{
  return ++m_cRef;
}
ULONG CSeqStream::Release(void)
{
// ASSERT(m_cRef);
  if(--m_cRef)
    return m_cRef;
  delete this;
  return 0;
}
HRESULT CSeqStream::QueryInterface(REFIID riid, void** ppv)
{
//
  ASSERT(ppv);
  *ppv = NULL;
```

```
if (riid == IID_IUnknown)
    *ppv = this;
  if (riid == IID_ISequentialStream)
    *ppv = this;
  if(*ppv)
  {
    ((IUnknown*)*ppv)->AddRef();
    return S_OK;
  }
  return E_NOINTERFACE;
}
BOOL CSeqStream::Seek(ULONG iPos)
{
  //Make sure the desired position is within the buffer
// ASSERT(iPos == 0 || iPos < m_cBufSize);</pre>
  //Reset the current buffer position
  m_iPos = iPos;
  return TRUE;
}
BOOL CSeqStream::Clear()
{
  //Frees the buffer
  m iPos
          = 0:
  m cBufSize = 0;
  CoTaskMemFree(m_pBuffer);
  m_pBuffer = NULL;
```

```
return TRUE;
```

```
}
```

```
BOOL CSeqStream::CompareData(void* pBuffer)
```

```
{
    ASSERT(pBuffer);
```

```
//Quick and easy way to compare user buffer with the stream
return memcmp(pBuffer, m_pBuffer, m_cBufSize)==0;
}
```

```
HRESULT CSeqStream::Read(void *pv, ULONG cb, ULONG* pcbRe
```

```
//Parameter checking
if(pcbRead)
 *pcbRead = 0;
```

```
if(!pv)
  return STG_E_INVALIDPOINTER;
```

```
if(cb == 0)
return S_OK;
```

```
//Actual code
ULONG cBytesLeft = m_cBufSize - m_iPos;
ULONG cBytesRead = cb > cBytesLeft ? cBytesLeft : cb;
```

```
//if no more bytes to retrieve return
if(cBytesLeft == 0)
return S_FALSE;
```

//Copy to users buffer the number of bytes requested or remaining memcpy(pv, (void\*)((BYTE\*)m\_pBuffer + m\_iPos), cBytesRead);

```
m_iPos += cBytesRead;
  if(pcbRead)
    *pcbRead = cBytesRead;
  if(cb != cBytesRead)
    return S_FALSE;
  return S_OK;
}
HRESULT CSeqStream::Write(const void *pv, ULONG cb, ULONG*
{
  //Parameter checking
  if(!pv)
    return STG_E_INVALIDPOINTER;
  if(pcbWritten)
    *pcbWritten = 0;
  if(cb == 0)
    return S_OK;
  //Enlarge the current buffer
  m_cBufSize += cb;
  //Need to append to the end of the stream
  m_pBuffer = CoTaskMemRealloc(m_pBuffer, m_cBufSize);
  memcpy((void*)((BYTE*)m_pBuffer + m_iPos), pv, cb);
  //m_iPos += cb;
  if(pcbWritten)
    *pcbWritten = cb;
```

```
return S_OK;
}
//....
void main()
{
CoInitialize(NULL);
```

```
DBOBJECT ObjectStruct;
ObjectStruct.dwFlags = STGM_READ;
ObjectStruct.iid = IID_ISequentialStream;
```

```
struct BLOBDATA
```

```
{
```

```
DBSTATUS dwStatus;
DWORD dwLength;
ISequentialStream* pISeqStream;
};
```

```
BLOBDATA BLOBGetData;
BLOBDATA BLOBSetData;
```

```
const ULONG cBindings = 1;
DBBINDING rgBindings[cBindings];
HRESULT hr = S OK;
             pIAccessor
                           = NULL;
IAccessor*
                pICommandText
ICommandText*
                                 = NULL;
ICommandProperties* pICommandProperties = NULL;
IRowsetChange* pIRowsetChange
                                = NULL;
                          = NULL;
IRowset*
             pIRowset
CSeqStream* pMySeqStream = NULL;
ULONG cRowsObtained = 0:
```

```
HACCESSOR hAccessor = DB_NULL_HACCESSOR;
DBBINDSTATUS rgBindStatus[cBindings];
HROW* rghRows = NULL;
const ULONG cPropSets = 1;
DBPROPSET rgPropSets[cPropSets];
const ULONG cProperties = 1;
DBPROP rgProperties[cProperties];
const ULONG cBytes = 10;
BYTE pBuffer[cBytes];
ULONG cBytesRead = 0;
```

```
BYTE pReadData[cBytes]; //read BLOB data in this array memset(pReadData, 0xAA, cBytes);
```

```
BYTE pWriteData[cBytes]; //write BLOB data from this array memset(pWriteData, 'D', cBytes);
```

```
goto Exit;
} //end if
```

/\*

```
Set the DBPROPSET structure. It is used to pass an array
of DBPROP structures to SetProperties().
*/
rgPropSets[0].guidPropertySet = DBPROPSET_ROWSET;
rgPropSets[0].cProperties = cProperties;
rgPropSets[0].rgProperties = rgProperties;
```

```
//Now set properties in the property group (DBPROPSET_ROWSE'
rgPropSets[0].rgProperties[0].dwPropertyID = DBPROP_UPDATA]
rgPropSets[0].rgProperties[0].dwOptions = DBPROPOPTIONS_RE
rgPropSets[0].rgProperties[0].dwStatus = DBPROPSTATUS_OK;
rgPropSets[0].rgProperties[0].colid = DB_NULLID;
rgPropSets[0].rgProperties[0].vValue.vt = VT_I4;
V_I4(&rgPropSets[0].rgProperties[0].vValue) = DBPROPVAL_UP_
```

```
if (FAILED(hr))
```

{

printf("Failed to get ICommandProperties to set rowset properties //Release any references and return.

goto Exit;

} //end if

```
hr = pICommandProperties->SetProperties(cPropSets, rgPropSets);
if (FAILED(hr))
```

{

printf("Execute failed to set rowset properties.\n");
//Release any references and return.

goto Exit; } //end if

```
//Execute a command (SELECT * FROM TestISeqStream)
hr = pICommandText->SetCommandText(DBGUID_DBSQL,
                                   L''SELECT * FROM TestISeqStream'');
if (FAILED(hr))
{
    printf("Failed to set command text SELECT * FROM.\n");
```

//Release any references and return.

goto Exit;

} //end if

```
if (FAILED(hr))
```

```
{
```

```
printf("Failed to execute the command SELECT * FROM.\n"); //Release any references and return.
```

goto Exit;

```
} //end if
```

```
//Fill the DBBINDINGS array.
rgBindings[0].iOrdinal = 2; //ordinal position
rgBindings[0].obValue = offsetof(BLOBDATA, pISeqStream);
rgBindings[0].obLength = offsetof(BLOBDATA, dwLength);
rgBindings[0].obStatus = offsetof(BLOBDATA, dwStatus);
rgBindings[0].pTypeInfo = NULL;
rgBindings[0].pObject = &ObjectStruct;
rgBindings[0].pBindExt = NULL;
rgBindings[0].dwPart = DBPART_VALUE | DBPART_STATUS | I
rgBindings[0].dwMemOwner = DBMEMOWNER_CLIENTOWNE
rgBindings[0].eParamIO = DBPARAMIO_NOTPARAM;
```

```
rgBindings[0].cbMaxLen = 0;
rgBindings[0].dwFlags = 0;
rgBindings[0].wType = DBTYPE_IUNKNOWN;
rgBindings[0].bPrecision = 0;
rgBindings[0].bScale = 0;
//Create an accessor using the binding information.
hr = pIRowsetChange->QueryInterface(IID_IAccessor,
                     (void**)&pIAccessor);
if (FAILED(hr))
{
  printf("Failed to get IAccessor interface.\n");
  //Release any references and return.
  goto Exit;
} //end if
hr = pIAccessor->CreateAccessor(DBACCESSOR_ROWDATA,
                   cBindings,
                   rgBindings,
                   sizeof(BLOBDATA),
                   &hAccessor,
                   rgBindStatus);
if (FAILED(hr))
{
  printf("Failed to create an accessor.\n");
  //Release any references and return.
  goto Exit;
} //end if
//Now get the first row.
hr = pIRowsetChange->QueryInterface(IID_IRowset,
                     (void **)&pIRowset);
if (FAILED(hr))
```

{

```
printf("Failed to get IRowset interface.\n");
  //Release any references and return.
  goto Exit;
} //end if
hr = pIRowset->GetNextRows(NULL,
                0,
                1,
                &cRowsObtained,
                &rghRows);
hr = pIRowset->GetData(rghRows[0],
             hAccessor,
             &BLOBGetData);
//Verify the retrieved data, only if data is not null.
if (BLOBGetData.dwStatus == DBSTATUS_S_ISNULL)
{
  //Process null data
  printf("Provider returned a null value.\n");
} else if(BLOBGetData.dwStatus == DBSTATUS_S_OK)
 //Provider returned a nonNULL value
{
  BLOBGetData.pISeqStream->Read(
                  pBuffer,
                  cBytes,
                  &cBytesRead);
  if(memcmp(pBuffer, pReadData, cBytes) != 0)
  {
    //cleanup
   }
```

```
SAFE_RELEASE(BLOBGetData.pISeqStream);
}
```

```
//Set up data for SetData.
pMySeqStream = new CSeqStream();
```

```
/*
Put data in to the ISequentialStream object
for the provider to write.
*/
pMySeqStream->Write(pWriteData,
cBytes,
NULL);
```

```
BLOBSetData.pISeqStream = (ISequentialStream*)pMySeqStream;
BLOBSetData.dwStatus = DBSTATUS_S_OK;
BLOBSetData.dwLength = pMySeqStream->Length();
```

```
//Release any references and return.
    goto Exit;
  } //end if
  hr = pIRowset->ReleaseRows(cRowsObtained,
                 rghRows,
                 NULL,
                 NULL,
                 NULL);
  if (FAILED(hr))
  {
    printf("Failed to release rows.\n");
    //Release any references and return.
    goto Exit;
  } //end if
Exit:
  //Free up all allocated memory and release interface pointers.
```

```
CoUninitialize();
```

} //end main.

```
//....
```

HRESULT GetCommandObject(REFIID riid, IUnknown\*\* ppIUnkno<sup>-</sup> {

HRESULT hr = S\_OK;

//Local interface pointers, until a connection is made. IDBInitialize\* pIDBInitialize = NULL; IDBProperties\* pIDBProperties = NULL; IDBCreateSession\* pIDBCreateSession = NULL; IDBCreateCommand\* pIDBCreateCommand = NULL;

```
const ULONG cPropSets = 1;
DBPROPSET rgPropSets[cPropSets];
```

```
const ULONG cProperties = 4;
DBPROP rgProperties[cProperties];
```

/\*

Initialize the property values needed to establish the connection.

\*/

for(ULONG i = 0; i < 4; i++)
VariantInit(&rgProperties[i].vValue);</pre>

```
//Server name.
```

```
rgProperties[0].dwPropertyID = DBPROP_INIT_DATASOURCE;
rgProperties[0].vValue.vt = VT_BSTR;
```

rgProperties[0].vValue.bstrVal =

```
SysAllocString(L"server");
rgProperties[0].dwOptions = DBPROPOPTIONS_REQUIRED;
```

```
rgProperties[0].colid = DB_NULLID;
```

```
rgProperties[2].colid = DB_NULLID;
```

## /\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method.

\*/

```
rgPropSets[0].guidPropertySet = DBPROPSET_DBINIT;
rgPropSets[0].cProperties = cProperties;
rgPropSets[0].rgProperties = rgProperties;
```

} //end if

```
//Set initialization properties.
hr = pIDBInitialize->QueryInterface(IID_IDBProperties,
                      (void **)&pIDBProperties);
if(FAILED(hr))
{
  printf("Failed to get IDBProperties interface.\n");
  //Release any references and return.
  goto Exit;
} //end if
hr = pIDBProperties->SetProperties(cPropSets, rgPropSets);
if(FAILED(hr))
{
  printf("Failed to set properties for DBPROPSET_DBINIT.\n");
  //Release any references and return.
  goto Exit;
} //end if
hr = pIDBInitialize->Initialize();
 if(FAILED(hr))
{
  printf("Failed to initialize.\n");
  //Release any references and return.
  goto Exit;
} //end if
//Create a session object.
 hr = pIDBInitialize->QueryInterface(
                 IID IDBCreateSession,
                 (void **)&pIDBCreateSession);
```

```
if(FAILED(hr))
```

{

```
printf("Failed to get pIDBCreateSession interface.\n");
    //Release any references and return.
    goto Exit;
  } //end if
    hr = pIDBCreateSession->CreateSession(
                     NULL,
                     IID IDBCreateCommand,
                     (IUnknown**)&pIDBCreateCommand);
  if(FAILED(hr))
  {
    printf("Failed to create session object.\n");
    //Release any references and return.
    goto Exit;
  } //end if
  //Get CommandText object
  hr = pIDBCreateCommand->CreateCommand(
                     NULL,
                     riid,
                     (IUnknown**)ppIUnknown);
  if(FAILED(hr))
  {
    printf("Failed to create CommandText object.\n");
    //Release any references and return.
    goto Exit;
  } //end if
  return hr;
Exit:
```

//Free up all allocated memory and release interface pointers. return hr;

```
} //end function
//....
HRESULT CreateTable(ICommandText* pICommandText)
{
  HRESULT hr = S_OK;
  //Drop existing table.
  hr = pICommandText->SetCommandText(
               DBGUID_DBSQL,
               L"DROP TABLE TestISeqStream");
  if(FAILED(hr))
  {
    printf("Failed to set command text DROP TABLE.\n");
    //Release any references and return.
    goto Exit;
  } //end if
  hr = pICommandText->Execute(NULL, IID_NULL, NULL, NULL,
  if(FAILED(hr))
  {
    printf("Failed to drop the table.\n");
    //Release any references and return.
    goto Exit;
  } //end if
  //Create a new table.
  hr = pICommandText->SetCommandText(DBGUID_DBSQL,
```

L"CREATE TABLE TestISeqStream (col1 int,col2 image)"); if(FAILED(hr))

{

printf("Failed to set command text CREATE TABLE.\n");

```
//Release any references and return.
goto Exit;
//ord if
```

} //end if

```
hr = pICommandText->Execute(NULL, IID_NULL, NULL, NULL,
if(FAILED(hr))
```

```
{
```

```
printf("Failed to create new table.\n");
//Release any references and return.
```

goto Exit;

} //end if

```
//Insert one row into table.
```

```
hr = pICommandText->SetCommandText(DBGUID_DBSQL,
L"INSERT INTO TestISeqStream(col1,col2) VALUES (1,0xAAAA
if(FAILED(hr))
```

{

```
printf("Failed to set command text INSERT INTO.\n");
//Release any references and return.
```

goto Exit;

```
} //end if
```

```
hr = pICommandText->Execute(NULL, IID_NULL, NULL, NULL,
if(FAILED(hr))
```

{

```
printf("Failed to insert record in the table.\n");
//Release any references and return.
goto Exit;
```

goto Exil

} //end if

Exit:

//Free up all allocated memory and release interface pointers.
return hr;

## } //end function

How To

## **Enumerating OLE DB Data Sources**

SQLOLEDB has an enumerator that a consumer can call to search for accessible data sources. Consumers should use enumerators to search for data sources, rather than searching the registry directly. In that way, the consumers will continue to work if the registry information changes.

How To

## How to enumerate OLE DB data sources (OLE DB)

To list the data sources visible to the SQLOLEDB enumerator, the consumer calls the **ISourcesRowset::GetSourcesRowset** method. This method returns a rowset of information about the currently visible data sources.

Depending on the network library used, the appropriate domain is searched for the data sources. For Named Pipes, it is the domain to which the client is logged on. For AppleTalk, it is the default zone. For SPX/IPX, it is the list of SQL Server installations found in the bindery. For Banyan VINES, it is the SQL Server installations found on the local network. Multiprotocol and TCP/IP sockets are not supported.

When the server is turned off or on, it can take few minutes to update the information in these domains.

#### To enumerate OLE DB data sources

- 1. Retrieve the source rowset by calling **ISourceRowset::GetSourcesRowset**.
- 2. Find the description of the enumerators rowset by calling **GetColumnInfo::IColumnInfo**.
- 3. Create the binding structures from the column information.
- 4. Create the rowset accessor by calling **IAccessor::CreateAccessor**.
- 5. Fetch the rows by calling **IRowset::GetNextRows**.
- 6. Retrieve data from the rowset's copy of the row by calling **IRowset::GetData** and process it.

//How to use the enumerator object to list
//the data sources available.

```
#define UNICODE
#define _UNICODE
#define DBINITCONSTANTS
#define INITGUID
```

```
#include <windows.h>
#include <stddef.h>
#include <oledb.h>
#include <oledberr.h>
#include <SQLOLEDB.h>
#include <stdio.h>
```

```
#define NUMROWS_CHUNK 5
```

```
//AdjustLen supports binding on four-byte boundaries.
_inline ULONG AdjustLen(ULONG cb)
{
    return ((cb + 3) & ~3);
}
```

// Get the characteristics of the rowset (the IColumnsInfo interface). HRESULT GetColumnInfo

```
(

IRowset* pIRowset,

UINT* pnCols,

DBCOLUMNINFO** ppColumnsInfo,

OLECHAR** ppColumnStrings

)

{

IColumnsInfo* pIColumnsInfo;

HRESULT hr;
```

}

// Create binding structures from column information. Binding structur
// will be used to create an accessor that allows row value retrieval.
void CreateDBBindings

```
(
UINT
           nCols,
                   pColumnsInfo,
DBCOLUMNINFO*
                ppDBBindings,
DBBINDING**
BYTE**
             ppRowValues
)
{
ULONG
             nCol;
             cbRow = 0:
ULONG
ULONG
             cbCol;
DBBINDING*
                pDBBindings;
```

```
BYTE* pRowValues;
```

```
pDBBindings = new DBBINDING[nCols];
for (nCol = 0; nCol < nCols; nCol++)
    {
        pDBBindings[nCol].iOrdinal = nCol+1;
        pDBBindings[nCol].pTypeInfo = NULL;
        pDBBindings[nCol].pObject = NULL;
        pDBBindings[nCol].pBindExt = NULL;
        pDBBindings[nCol].dwPart = DBPART_VALUE;
        pDBBindings[nCol].dwMemOwner = DBMEMOWNER_CLIEN
        pDBBindings[nCol].eParamIO = DBPARAMIO_NOTPARAM;
        pDBBindings[nCol].eParamIO = DBPARAMIO_NOTPARAM;
        pDBBindings[nCol].wType = pColumnsInfo[nCol].wType;
        pDBBindings[nCol].bPrecision = pColumnsInfo[nCol].bPrecisior
        pDBBindings[nCol].bScale = pColumnsInfo[nCol].bScale;
```

```
cbCol = pColumnsInfo[nCol].ulColumnSize;
```

```
switch (pColumnsInfo[nCol].wType)
{
    case DBTYPE_STR:
        {
            cbCol += 1;
            break;
        }
    case DBTYPE_WSTR:
        {
            cbCol = (cbCol + 1) * sizeof(WCHAR);
            break;
        }
    }
}
```

```
default:
        break;
      }
    pDBBindings[nCol].obValue = cbRow;
    pDBBindings[nCol].cbMaxLen = cbCol;
    cbRow += AdjustLen(cbCol);
   }
  pRowValues = new BYTE[cbRow];
  *ppDBBindings = pDBBindings;
  *ppRowValues = pRowValues;
  return;
  }
int main()
  ISourcesRowset*
                    pISourceRowset = NULL;
  IRowset*
                 pIRowset = NULL;
  IAccessor*
                 pIAccessor = NULL;
                     pDBBindings = NULL;
  DBBINDING*
  HROW*
                  pRows = new HROW[500];
  BYTE*
                 pData = NULL;
  HACCESSOR
                     hAccessorRetrieve = NULL;
  ULONG
                  cRows = 0;
  ULONG
                  DSSeqNumber = 0;
  HRESULT
                   hr;
                nCols;
  UINT
                         pColumnsInfo = NULL;
  DBCOLUMNINFO*
```

{

OLECHAR*	pColumnStrings = NULL;
DBBINDSTATU	JS* pDBBindStatus = NULL;
BYTE*	pRowValues = NULL;
ULONG	cRowsObtained;
ULONG	iRow;
char*	pMultiByte = NULL;
short*	psSourceType = NULL;
BYTE*	pDatasource = NULL;

//Initialize COM library.
CoInitialize(NULL);

```
//Initialize the enumerator.
if(FAILED(CoCreateInstance(CLSID_SQLOLEDB_ENUMERATC
               NULL,
               CLSCTX_INPROC_SERVER,
               IID_ISourcesRowset,
               (void**)&pISourceRowset)))
{
  //Process error.
  return TRUE;
}
//Retrieve the source rowset.
hr = pISourceRowset->GetSourcesRowset(NULL,
                    IID_IRowset,
                    0,
                    NULL,
                    (IUnknown**)&pIRowset);
pISourceRowset->Release();
if(FAILED(hr))
{
  //Process error.
```

```
return TRUE;
}
//Get the description of the enumerator's rowset.
if(FAILED(hr = GetColumnInfo(pIRowset,
                &nCols,
                &pColumnsInfo,
                &pColumnStrings)))
{
  //Process error.
  goto SAFE_EXIT;
}
//Create the binding structures.
CreateDBBindings(nCols,
         pColumnsInfo,
         &pDBBindings,
         &pRowValues);
pDBBindStatus = new DBBINDSTATUS[nCols];
if (sizeof(TCHAR) != sizeof(WCHAR))
{
  pMultiByte = new char[pDBBindings[0].cbMaxLen];
}
if(FAILED(pIRowset->QueryInterface(IID_IAccessor, (void**)&pI
{
  //Process error.
  goto SAFE_EXIT;
}
//Create the rowset accessor.
if(FAILED(hr = pIAccessor->CreateAccessor(DBACCESSOR_RO)
                       nCols,
                       pDBBindings,
                       0,
```

```
&hAccessorRetrieve,
                       pDBBindStatus)))
{
  //Process error.
  goto SAFE_EXIT;
}
//Process all the rows, NUMROWS_CHUNK rows at a time.
while (SUCCEEDED(hr))
ł
hr=pIRowset->GetNextRows(NULL,
              0,
              NUMROWS_CHUNK,
              &cRowsObtained,
              &pRows);
if(FAILED(hr))
ł
  //process error
}
if(cRowsObtained == 0 \parallel FAILED(hr))
  break;
  for(iRow = 0; iRow < cRowsObtained; iRow++)</pre>
  {
    //Get the rowset data.
    if(SUCCEEDED(hr = pIRowset->GetData(pRows[iRow],
                        hAccessorRetrieve,
                        pRowValues)))
    {
    psSourceType = (short *)(pRowValues +
                  pDBBindings[3].obValue);
    if (*psSourceType == DBSOURCETYPE_DATASOURCE)
```

```
{
    DSSeqNumber = DSSeqNumber + 1; //Data source counter.
    pDatasource = (pRowValues + pDBBindings[0].obValue);
    if(sizeof(TCHAR) != sizeof(WCHAR))
    {
      WideCharToMultiByte(CP_ACP, 0,
        (WCHAR*)pDatasource, -1, pMultiByte,
        pDBBindings[0].cbMaxLen, NULL, NULL);
      printf( "DataSource# %d\tName: %S\n",
               DSSeqNumber, (WCHAR *) pMultiByte );
    }
    else
    {
      printf( "DataSource# %d\tName: %S\n",
           DSSeqNumber, (WCHAR *) pDatasource );
    } //if
  } //if
} //if
```

```
} //for
```

pIRowset->ReleaseRows(cRowsObtained, pRows, NULL, NULL } //while //Release COM library. CoUninitialize();

return(0); SAFE\_EXIT: //Do the clean-up. return TRUE; }; How To

# **Bulk-Copying Rowsets**

SQLOLEDB implements the provider-specific **IRowsetFastLoad** interface to expose support for Microsoft® SQL Server<sup>™</sup> bulk copying from a consumer to a SQL Server table.

How To

# How to bulk copy data using IRowsetFastLoad (OLE DB)

The consumer notifies SQLOLEDB of its need for bulk copying by setting the SQLOLEDB provider-specific property **SSPROP\_ENABLEFASTLOAD** to VARIANT\_TRUE. With the property set on the data source, the consumer creates a SQLOLEDB session. The new session allows the consumer access to **IRowsetFastLoad**.

### To bulk copy data into a SQL Server table

- 1. Establish a connection to the data source.
- 2. Set the SQLOLEDB provider-specific data source property **SSPROP\_ENABLEFASTLOAD** to VARIANT\_TRUE. With this property set to VARIANT\_TRUE, the newly created session allows the consumer access to **IRowsetFastLoad**.
- 3. Create a session requesting the **IOpenRowset** interface.
- 4. Call **IOpenRowset::OpenRowset** to open a rowset that includes all the rows from the table (in which data is to be copied using bulk-copy operation).
- 5. Do the necessary bindings and create an accessor using **IAccessor::CreateAccessor**.
- 6. Set up the memory buffer from which the data will be copied to the table.
- 7. Call **IRowsetFastLoad::InsertRow** to bulk copy the data in to the table.

The following example illustrates the use of **IRowsetFastLoad** for bulk copying of the records into a table. In this example, 10 records will be added to the table IRFLTable. You need to create the table **IRFLTable** in the database.

CREATE TABLE IRFLTable (col\_vchar varchar(30))

#define DBINITCONSTANTS

#include <oledb.h>
#include <oledberr.h>
#include <stdio.h>
#include <stddef.h> //for offsetof
#include <sqloledb.h>

/\* @type UWORD | 2 byte unsigned integer. \*/
typedef unsigned short UWORD;
/\* @type SDWORD | 4 byte signed integer. \*/
typedef signed long SDWORD;

```
WCHAR g_wszTable[] = L"IRFLTable";
WCHAR g_strTestLOC[100] = L"server ";
WCHAR g_strTestDSN[] = L"database";
WCHAR g_strTestUID[] = L"login";
WCHAR g_strTestPWD[] = L"password";
const UWORD g_cOPTION = 4;
const UWORD MAXPROPERTIES = 5;
const ULONG DEFAULT_CBMAXLENGTH = 20;
IMalloc* g_pIMalloc = NULL;
IDBInitialize* g_pIDBInitialize = NULL;
```

/\*

Given an ICommand pointer, properties, and query, a rowsetpointer is returned.

```
HRESULT CreateSessionCommand
(
DBPROPSET* rgPropertySets,
ULONG ulcPropCount,
CLSID clsidProv
);
```

//Use to set properties and execute a given query. HRESULT ExecuteQuery

```
(
IDBCreateCommand* pIDBCreateCommand,
WCHAR* pwszQuery,
DBPROPSET* rgPropertySets,
ULONG ulcPropCount,
LONG* pcRowsAffected,
IRowset** ppIRowset,
BOOL fSuccessOnly = TRUE
);
```

//Use to set up options for call to IDBInitialize::Initialize. void SetupOption

```
(
DBPROPID PropID,
WCHAR *wszVal,
DBPROP * pDBProp
);
```

//Sets fastload property on/off for session.
HRESULT SetFastLoadProperty(BOOL fSet);

//IRowsetFastLoad inserting data.
HRESULT FastLoadData(void);

// How to lay out each column in memory.
struct COLUMNDATA {

SDWORDdwLength; // Length of data (not space allocated).DWORDdwStatus; // Status of column.BYTEbData[1]; // Store data here as a variant.};

```
COLUMN ALIGNVAL
#define
                                  8
#define ROUND_UP(Size, Amount)(((DWORD)(Size) + ((Amount)-1
int main()
  {
  HRESULT
                  hr = NOERROR;
                  hr2 = NOERROR:
  HRESULT
                fResults = FALSE:
  BOOL
            //OLE initialized?
                fInitialized = FALSE;
  BOOL
            //One property set for initializing.
                    rgPropertySets[1];
  DBPROPSET
            //Properties within above property set.
                 rgDBProperties[g_cOPTION];
  DBPROP
            //Property count.
                 ulPropCount = 0;
  ULONG
            //# of initialization properties.
                 cOptions = g_cOPTION;
  ULONG
   IDBCreateCommand* pIDBCreateCommand = NULL;
  IRowset*
                pIRowset
                               = NULL;
  DBPROPSET*
                     rgProperties
                                   = NULL:
```

IAccessor\* pIAccessor = NULL;

//Basic initialization.

```
if(FAILED(CoInitialize(NULL)))
  goto cleanup;
else
  fInitialized = TRUE;
hr = CoGetMalloc(MEMCTX_TASK, &g_pIMalloc);
if((!g_pIMalloc) || FAILED(hr))
  {
  goto cleanup;
  ł
/*
Set up property set for call to IDBInitialize
in CreateSessionCommand.
*/
rgPropertySets[0].rgProperties = rgDBProperties;
rgPropertySets[0].cProperties = g_cOPTION;
rgPropertySets[0].guidPropertySet = DBPROPSET DBINIT;
SetupOption(DBPROP_INIT_CATALOG,
       g_strTestDSN,
       &rgDBProperties[0]);
SetupOption(DBPROP_AUTH_USERID,
      g strTestUID,
      &rgDBProperties[1]);
SetupOption(DBPROP_AUTH_PASSWORD,
      g_strTestPWD,
      &rgDBProperties[2]);
SetupOption(DBPROP_INIT_DATASOURCE,
      g_strTestLOC,
      &rgDBProperties[3]);
```

if(!SUCCEEDED(hr=CreateSessionCommand(rgPropertySets,

```
1,
CLSID_SQLOLEDB)))
```

goto cleanup;

```
//Get IRowsetFastLoad and insert data into IRFLTable.
if(FAILED(hr = FastLoadData()))
goto cleanup;
```

cleanup:

```
//Free memory.
if(rgProperties && rgProperties->rgProperties)
    delete [](rgProperties->rgProperties);
if(rgProperties)
    delete []rgProperties;
if(pIDBCreateCommand)
    pIDBCreateCommand->Release();
```

```
if(pIAccessor)
    pIAccessor->Release();
```

```
if(pIRowset)
    pIRowset->Release();
if(g_pIMalloc)
    g_pIMalloc->Release();
```

```
if(g_pIDBInitialize)
{
    hr2 = g_pIDBInitialize->Uninitialize();
    if(FAILED(hr2))
        printf("Uninitialize failed\n");
}
```

```
if(fInitialized)
```

```
CoUninitialize();
```

```
if(SUCCEEDED(hr))
   printf("Test completed successfully.\n\n");
 else
   printf("Test failed.\n\n");
 return(0);
  }
//-----
HRESULT FastLoadData(void)
{
 HRESULT
                   = E FAIL;
              hr
              hr2 = E_FAIL;
 HRESULT
 DBID
            TableID;
 IDBCreateSession* pIDBCreateSession = NULL;
 IOpenRowset* pIOpenRowsetFL = NULL;
 IRowsetFastLoad* pIFastLoad = NULL;
 IAccessor* pIAccessor
                         = NULL;
 HACCESSOR
                hAccessor
                         = 0:
 DBBINDSTATUS
                  oneStatus
                          = 0;
 DBBINDING
                oneBinding;
                        = 0;
 ULONG
             ulOffset
 TableID.uName.pwszName
                            = NULL;
             i
                 = 0;
 LONG
           pData
 void*
                    = NULL;
 COLUMNDATA*
                  pcolData
                             = NULL;
             strData[] = "Show me the money!";
 CHAR
```

TableID.eKind = DBKIND\_NAME; TableID.uName.pwszName = new WCHAR[wcslen(g\_wszTable)+2 wcscpy(TableID.uName.pwszName, g\_wszTable);

//Next set up an accessor for the data.

```
oneBinding.pTypeInfo = NULL;
oneBinding.obValue = ulOffset + offsetof(COLUMNDATA,bData)
oneBinding.obLength = ulOffset + offsetof(COLUMNDATA,dwLe
oneBinding.obStatus = ulOffset + offsetof(COLUMNDATA,dwStat
oneBinding.cbMaxLen = 30; //Size of varchar column.
oneBinding.pTypeInfo = NULL;
oneBinding.pObject = NULL;
oneBinding.pBindExt = NULL;
oneBinding.dwFlags = 0;
oneBinding.eParamIO = DBPARAMIO NOTPARAM;
oneBinding.dwMemOwner = DBMEMOWNER_CLIENTOWNED
oneBinding.bPrecision= 0;
oneBinding.bScale = 0;
oneBinding.wType = DBTYPE STR;
ulOffset = oneBinding.cbMaxLen + offsetof(COLUMNDATA, bDat
ulOffset = ROUND_UP( ulOffset, COLUMN_ALIGNVAL );
```

return hr;

```
if(FAILED(hr = pIAccessor->CreateAccessor(
DBACCESSOR_ROWDATA,
1,
&oneBinding,
ulOffset,
&hAccessor,
&oneStatus)))
```

return hr;

//Set up memory buffer.
pData = new BYTE[40];

```
pcolData = (COLUMNDATA*)pData;
  pcolData->dwLength = strlen("Show the data") + 1;
  pcolData->dwStatus = 0;
  memcpy(&(pcolData->bData), "Show the data",
                strlen("Show me data") + 1);
  for(i=0; i<10; i++)
  {
    if(FAILED(hr = pIFastLoad->InsertRow(hAccessor, pData)))
      goto cleanup;
  }
  if(FAILED(hr = pIFastLoad->Commit(TRUE)))
    printf("Error on IRFL::Commit\n");
cleanup:
  if(FAILED(hr2 = SetFastLoadProperty(FALSE)))
  {
    printf("SetFastLoadProperty(FALSE) failed with %x", hr2);
  }
  if(pIAccessor && hAccessor)
  {
    if(FAILED(pIAccessor->ReleaseAccessor(hAccessor, NULL)))
      hr = E_FAIL;
  }
  if(pIAccessor)
    pIAccessor->Release();
  if(pIFastLoad)
    pIFastLoad->Release();
  if(pIOpenRowsetFL)
    pIOpenRowsetFL->Release();
  if(pIDBCreateSession)
```

```
pIDBCreateSession->Release();
```

```
if(TableID.uName.pwszName)
  delete []TableID.uName.pwszName;
```

```
return hr;

}

//------

HRESULT SetFastLoadProperty(BOOL fSet)

{

HRESULT hr = S_OK;

IDBProperties * pIDBProps = NULL;

DBPROP rgProps[1];

DBPROPSET PropSet;
```

```
VariantInit(&rgProps[0].vValue);
```

```
rgProps[0].dwOptions = DBPROPOPTIONS_REQUIRED;
rgProps[0].colid = DB_NULLID;
rgProps[0].vValue.vt = VT_BOOL;
rgProps[0].dwPropertyID = SSPROP_ENABLEFASTLOAD;
if(fSet == TRUE)
rgProps[0].vValue.boolVal = VARIANT_TRUE;
else
rgProps[0].vValue.boolVal = VARIANT_FALSE;
PropSet.rgProperties = rgProps;
PropSet.cProperties = 1;
PropSet.guidPropertySet = DBPROPSET_SQLSERVERDATASC
```

```
if(SUCCEEDED(hr = g_pIDBInitialize->QueryInterface(
```

```
IID_IDBProperties,
                  (LPVOID *)&pIDBProps)))
    {
      hr = pIDBProps->SetProperties(1, &PropSet);
    }
    VariantClear(&rgProps[0].vValue);
    if(pIDBProps)
      pIDBProps->Release();
    return hr;
}
//_____
HRESULT CreateSessionCommand
  (
  DBPROPSET* rgPropertySets, //@parm [in] property sets.
  ULONG
            ulcPropCount, //@parm [in] count of prop sets.
  CLSID
           clsidProv //@parm [in] Provider CLSID.
  )
  {
  HRESULT hr = NOERROR;
  IDBCreateSession* pIDBCreateSession = NULL;
  IDBProperties* pIDBProperties = NULL;
  UWORD
                 i=0, j=0; //indexes.
  if(ulcPropCount && !rgPropertySets)
    hr = E_INVALIDARG;
    return hr;
    }
```

```
if (!SUCCEEDED(hr = CoCreateInstance(clsidProv,
                    NULL, CLSCTX_INPROC_SERVER,
                    IID IDBInitialize,
                    (void **)&g_pIDBInitialize)))
  {
    goto CLEANUP;
  }
  if (!SUCCEEDED(hr = g_pIDBInitialize->QueryInterface(
                    IID_IDBProperties,
                    (void **)&pIDBProperties)))
  {
    goto CLEANUP;
  }
  if (!SUCCEEDED(hr = pIDBProperties->SetProperties(
                       ulcPropCount,
                       rgPropertySets)))
  {
    goto CLEANUP;
  }
  if (!SUCCEEDED(hr = g_pIDBInitialize->Initialize()))
  {
    printf("Call to initialize failed.\n");
    goto CLEANUP;
  }
CLEANUP:
  if(pIDBProperties)
    pIDBProperties->Release();
  if(pIDBCreateSession)
    pIDBCreateSession->Release();
```

```
for(i = 0; i < ulcPropCount; i++)</pre>
    for(j = 0; j < rgPropertySets[i].cProperties; j++)</pre>
    ł
    VariantClear(&(rgPropertySets[i].rgProperties[j]).vValue);
    }
  return hr;
  }
//-----
void SetupOption
  (
  DBPROPID PropID,
  WCHAR *wszVal,
  DBPROP * pDBProp
  )
  {
  pDBProp->dwPropertyID = PropID;
  pDBProp->dwOptions = DBPROPOPTIONS_REQUIRED;
  pDBProp->colid = DB_NULLID;
  pDBProp->vValue.vt = VT_BSTR;
  pDBProp->vValue.bstrVal = SysAllocStringLen(
                 wszVal,
                 wcslen(wszVal));
  }
```

# **Obtaining a FAST\_FORWARD cursor**

Consumers can request different cursor behaviors in a rowset by setting certain rowset properties. If the consumer does not set any of these rowset properties, or sets them all to their default values, SQLOLEDB implements the rowset using a default result set. If any one of these properties are set to a value other than the default, SQLOLEDB implements the rowset using server cursors.

### How to obtain FAST\_FORWARD cursor

To obtain a forward-only, read-only cursor, set the rowset properties, DBPROP\_SERVERCURSOR, DBPROP\_OTHERINSERT, DBPROP\_OTHERUPDATEDELETE, DBPROP\_OWNINSERT, DBPROP\_OWNUPDATEDELETE to VARIANT\_TRUE.

### To obtain FAST\_FORWARD cursor

- 1. Establish a connection to the data source.
- Set the rowset properties, DBPROP\_SERVERCURSOR, DBPROP\_OTHERINSERT,
   DBPROP\_OTHERUPDATEDELETE, DBPROP\_OWNINSERT,
   DBPROP\_OWNUPDATEDELETE should be set to VARIANT\_TRUE
- 3. Execute the command.

The following example shows how to set the rowset properties to obtain a FAST\_FORWARD cursor. After the properties are set, a SELECT statement is executed to find the first and last names of authors in the **pubs** database.

#define INITGUID #define DBINITCONSTANTS

#include <windows.h>
#include <stdio.h>
#include <oledb.h>
#include <sqloledb.h>
#include <sqloledb.h>
#include <oledberr.h>

IDBInitialize\* pIDBInitialize = NULL; ICommandText\* pICommandText = NULL; // Connect to the server and create a command object.
void InitializeAndConnect();

// Set the properties to get a FAST\_FORWARD cursor.
void SetRowsetProperties();

// This function executes a command and displays the results.
void ExecuteAndDisplay();

```
// Clean up the memory.
void Cleanup();
```

```
void main()
```

```
{
```

```
// Initialize.
InitializeAndConnect();
```

// Set the row properties to FAST\_FORWARD cursor.
SetRowsetProperties();

```
// Execute a command and display the results.
ExecuteAndDisplay();
```

```
// Cleanup.
Cleanup();
```

```
void InitializeAndConnect()
```

```
{
```

}

```
HRESULT hr = S_OK;
IDBProperties* pIDBProperties = NULL;
IDBCreateSession* pIDBCreateSession = NULL;
IDBCreateCommand* pIDBCreateCommand = NULL;
```

```
DBPROPSET
                  dbPropSet;
                dbProp[4];
DBPROP
// Initialize OLE
if( FAILED( hr = OleInitialize( NULL ) ) )
{
  // Handle errors here.
}
// Create an instance of Microsoft OLE DB Provider for SQL Server
if( FAILED( hr = CoCreateInstance(
               CLSID_SQLOLEDB,
               NULL.
               CLSCTX INPROC SERVER,
               IID_IDBProperties,
               (void **) &pIDBProperties ) ) )
{
  // Handle errors here.
}
// Set up the connection properties.
dbProp[0].dwPropertyID = DBPROP_INIT_DATASOURCE;
dbProp[0].dwOptions = DBPROPOPTIONS_REQUIRED;
dbProp[0].colid
                    = DB NULLID;
V_VT(&(dbProp[0].vValue)) = VT_BSTR;
V_BSTR(&(dbProp[0].vValue)) = SysAllocString( L"server ");
dbProp[1].dwPropertyID = DBPROP_AUTH_USERID;
dbProp[1].dwOptions
                       = DBPROPOPTIONS REQUIRED;
dbProp[1].colid
                    = DB _NULLID;
V VT(&(dbProp[1].vValue)) = VT BSTR;
V BSTR(&(dbProp[1].vValue)) = SysAllocString( L"login");
```

```
dbProp[2].dwPropertyID = DBPROP_AUTH PASSWORD;
                    = DBPROPOPTIONS_REQUIRED;
dbProp[2].dwOptions
dbProp[2].colid
                    = DB NULLID;
V_VT(&(dbProp[2].vValue)) = VT_BSTR;
V BSTR(&(dbProp[2].vValue)) = SysAllocString( L''');
dbProp[3].dwPropertyID = DBPROP INIT CATALOG;
dbProp[3].dwOptions = DBPROPOPTIONS_REQUIRED;
                    = DB_NULLID;
dbProp[3].colid
V_VT(&(dbProp[3].vValue)) = VT_BSTR;
V_BSTR(&(dbProp[3].vValue)) = SysAllocString( L"pubs" );
dbPropSet.rgProperties = dbProp;
dbPropSet.cProperties
                       = 4:
dbPropSet.guidPropertySet = DBPROPSET_DBINIT;
if( FAILED( hr = pIDBProperties->SetProperties(
                   1,
                   &dbPropSet )))
{
  // Handle errors here.
}
SysFreeString( V_BSTR(&(dbProp[0].vValue)) );
SysFreeString( V_BSTR(&(dbProp[1].vValue)) );
SysFreeString( V_BSTR(&(dbProp[2].vValue)) );
SysFreeString( V_BSTR(&(dbProp[3].vValue)) );
// Get an IDBInitialize interface.
if( FAILED( hr = pIDBProperties->QueryInterface(
                IID IDBInitialize,
                (void **) &pIDBInitialize )))
{
```

```
// Handle errors here.
}
// Call Initialize.
if( FAILED( hr = pIDBInitialize->Initialize()))
{
  // Handle errors here.
}
// Get a IDBCreateSession interface.
if( FAILED( hr = pIDBInitialize->QueryInterface(
                IID_IDBCreateSession,
                (void **) &pIDBCreateSession )))
{
  // Handle errors here.
}
// Create a session
if( FAILED( hr = pIDBCreateSession->CreateSession(
                 NULL,
                 IID_IDBCreateCommand,
                 (IUnknown **) &pIDBCreateCommand)))
{
  // Handle errors here.
}
// Create a command.
if( FAILED( hr = pIDBCreateCommand->CreateCommand(
                   NULL,
                   IID_ICommandText,
                   (IUnknown **) &pICommandText)))
{
  // Handle errors here.
```

}

}

{

```
// Release all the objects not needed anymore.
  pIDBProperties->Release();
  pIDBCreateSession->Release();
  pIDBCreateCommand->Release();
void SetRowsetProperties()
  HRESULT
                  hr
                               = S OK:
  ICommandProperties* pICommandProperties = NULL;
  DBPROPSET
                    dbPropSet;
                  dbProp[5];
  DBPROP
  // Get an ICommandProperties object.
  if( FAILED( hr = pICommandText->QueryInterface(
            IID ICommandProperties,
            (void **) &pICommandProperties )))
  {
    // Handle errors here.
  }
  // Set up the properties to get a FAST_FORWARD cursor.
  dbProp[0].dwPropertyID = DBPROP_SERVERCURSOR;
  dbProp[0].dwOptions
                         = DBPROPOPTIONS REQUIRED;
  dbProp[0].colid
                      = DB NULLID;
  V_VT(&(dbProp[0].vValue)) = VT_BOOL;
  V BOOL(&(dbProp[0].vValue)) = VARIANT TRUE;
  dbProp[1].dwPropertyID = DBPROP OTHERINSERT;
  dbProp[1].dwOptions
                         = DBPROPOPTIONS_REQUIRED;
  dbProp[1].colid
                      = DB NULLID;
```

```
V_VT(&(dbProp[1].vValue)) = VT_BOOL;
V_BOOL(&(dbProp[1].vValue)) = VARIANT_TRUE;
```

dbProp[2].dwPropertyID = DBPROP\_OTHERUPDATEDELET dbProp[2].dwOptions = DBPROPOPTIONS\_REQUIRED; dbProp[2].colid = DB\_NULLID; V\_VT(&(dbProp[2].vValue)) = VT\_BOOL; V\_BOOL(&(dbProp[2].vValue)) = VARIANT\_TRUE;

```
dbProp[3].dwPropertyID = DBPROP_OWNINSERT;
dbProp[3].dwOptions = DBPROPOPTIONS_REQUIRED;
dbProp[3].colid = DB_NULLID;
V_VT(&(dbProp[3].vValue)) = VT_BOOL;
V BOOL(&(dbProp[3].vValue)) = VARIANT TRUE;
```

```
dbProp[4].dwPropertyID = DBPROP_OWNUPDATEDELETE;
dbProp[4].dwOptions = DBPROPOPTIONS_REQUIRED;
dbProp[4].colid = DB_NULLID;
V_VT(&(dbProp[4].vValue)) = VT_BOOL;
V_BOOL(&(dbProp[4].vValue)) = VARIANT_TRUE;
```

```
dbPropSet.rgProperties = dbProp;
dbPropSet.cProperties = 5;
dbPropSet.guidPropertySet = DBPROPSET_ROWSET;
```

// Handle errors here.

}

{

// Release the ICommandProperties object.

```
pICommandProperties->Release();
}
void ExecuteAndDisplay()
{
  HRESULT
                             = S OK;
                  hr
                pIRowset
  IRowset*
                             = NULL;
                pIAccessor = NULL;
  IAccessor*
  BYTE*
                             = NULL;
                pData
                 cRowsObtained
  ULONG
                                  = 0;
  ULONG
                 cCount
                               = 0;
  HROW*
                 pRows
                               = new HROW[10];
  HACCESSOR
                    hAccessor;
                   Bind<sup>[2]</sup>;
  DBBINDING
  // Set the command text.
  if( FAILED( hr = pICommandText->SetCommandText(
            DBGUID SQL,
            L"select au_lname, au_fname from authors")))
  {
    // Handle errors here.
  }
  // Execute the command.
  if( FAILED( hr = pICommandText->Execute(
                      NULL,
                      IID_IRowset,
                      NULL,
                      NULL,
                      (IUnknown **) &pIRowset )))
  {
```

// Handle errors here.

```
}
```

// Set up the binding structure for au\_lname (varchar(40)). Bind[0].dwPart = DBPART VALUE; = DBPARAMIO NOTPARAM; Bind[0].eParamIO Bind[0].iOrdinal = 1; Bind[0].pTypeInfo = NULL; Bind[0].pObject = NULL: Bind[0].pBindExt = NULL; Bind[0].dwFlags = 0: Bind[0].dwMemOwner = DBMEMOWNER CLIENTOWNED; Bind[0].obLength = 0;Bind[0].obStatus = 0:Bind[0].obValue = 0: Bind[0].cbMaxLen = 40;Bind[0].wType = DBTYPE\_STR; Bind[0].bPrecision = 0; Bind[0].bScale = 0:

// Set up the binding structure for au\_fname (varchar(20)). Bind[1].dwPart = DBPART VALUE; Bind[1].eParamIO = DBPARAMIO NOTPARAM; Bind[1].iOrdinal = 2; Bind[1].pTypeInfo = NULL; Bind[1].pObject = NULL; Bind[1].pBindExt = NULL; Bind[1].dwFlags = 0: Bind[1].dwMemOwner = DBMEMOWNER\_CLIENTOWNED; Bind[1].obLength = 0:Bind[1].obStatus = 0;Bind[1].obValue = 50; Bind[1].cbMaxLen = 20; Bind[1].wType = DBTYPE STR; Bind[1].bPrecision = 0;

```
Bind[1].bScale
                 = 0;
// Get an IAccessor interface.
if( FAILED( hr = pIRowset->QueryInterface(
                IID IAccessor,
                (void **) &pIAccessor)))
{
  // Handle errors here.
}
// Create an accessor.
if( FAILED( hr = pIAccessor->CreateAccessor(
                DBACCESSOR_ROWDATA,
                2,
                Bind,
                0,
                &hAccessor,
                NULL)))
{
  // Handle errors here.
}
// Allocate memory for the data.
pData = new BYTE[100];
// Loop through all of the rows.
while(TRUE)
{
  if( FAILED( hr = pIRowset->GetNextRows(
                  NULL,
                  0,
                   10.
                  &cRowsObtained,
```

```
&pRows)))
{
  // Handle errors here.
}
// Make sure some rows were obtained.
if( cRowsObtained == 0 )
{
  break;
}
// Get the data for the each of the rows.
for( cCount = 0; cCount < cRowsObtained; cCount++ )</pre>
{
  // Get the row data needed.
  if( FAILED( hr = pIRowset->GetData(
                   pRows[cCount],
                   hAccessor,
                   pData )))
  {
    // Handle errors here.
  }
  // Display row data.
  printf( "%s, %s\n", pData, ( pData + 50 ));
}
// Release the rows.
if( FAILED( hr = pIRowset->ReleaseRows(
                 cRowsObtained,
                 pRows,
                 NULL,
                 NULL,
```

```
NULL )))
{
// Handle errors here.
}
}
```

// Free the memory allocated for the data.
delete [] pData;

```
// Release the HACCESSOR.
  if( FAILED( hr = pIAccessor->ReleaseAccessor(
                        hAccessor,
                        NULL )))
  {
    // Handle errors here.
  }
  // Release the IAccessor object.
  pIAccessor->Release();
  // Release the rowset.
  pIRowset->Release();
}
void Cleanup()
{
  HRESULT hr = S_OK;
  // Release the ICommandText object.
  pICommandText->Release();
  // Uninitialize the IDBInitialize object.
```

```
if( FAILED( hr = pIDBInitialize->Uninitialize() ) )
```

```
{
    // Handle errors here.
}
// Release the IDBInitialize object.
pIDBInitialize->Release();
// Uninitialize OLE.
OleUninitialize();
```

}

# **Using Bookmarks**

Bookmarks allow consumers to return quickly to a row. The bookmark column is the column 0 in the rowset.

## How to retrieve rows using bookmarks (OLE DB)

The consumer sets the *dwFlag* field value of the binding structure to DBCOLUMNSINFO\_ISBOOKMARK to indicate that the column is used as bookmark. The consumer also sets the rowset property

**DBPROP\_BOOKMARKS** to VARIANT\_TRUE. This allows column 0 to be present in the rowset. **IRowsetLocate::GetRowsAt** is then used to fetch rows starting with the row specified an an offset from a bookmark.

### To retrieve rows using bookmarks

- 1. Establish a connection to the data source.
- 2. Set the rowset property **DBPROP\_IRowsetLocate** property to VARIANT\_TRUE.
- 3. Execute the command.
- 4. Set the *dwFlags* field of the binding structure to **DBCOLUMNSINFO\_ISBOOKMARK** flag for the column that will be used as a bookmark.
- 5. Use **IRowsetLocate::GetRowsAt** to fetch rows, starting with the row specified by an offset from the bookmark.

The following example shows how to fetch rows using a bookmark. In this example, the5th row is retrieved from the result set produced from the execution of a SELECT statement.

/\*

```
How to use bookmarks
*/
void InitializeAndEstablishConnection();
void ProcessResultSet();
```

```
#define UNICODE
#define _UNICODE
#define DBINITCONSTANTS
#define INITGUID
```

```
#include <stdio.h>
#include <tchar.h>
#include <stddef.h>
#include <windows.h>
#include <iostream.h>
#include <oledb.h>
#include <sqloledb.h>
```

```
pIDBInitialize = NULL;
IDBInitialize*
                pIDBProperties
IDBProperties*
                               = NULL;
IDBCreateSession*
                  pIDBCreateSession = NULL;
IDBCreateCommand* pIDBCreateCommand = NULL;
ICommandProperties* pICommandProperties = NULL;
ICommandText*
                 pICommandText = NULL;
IRowset*
              pIRowset
                           = NULL;
IColumnsInfo*
                pIColumnsInfo
                                = NULL;
DBCOLUMNINFO*
                     pDBColumnInfo = NULL;
              pIAccessor
                            = NULL;
IAccessor*
IRowsetLocate*
                pIRowsetLocate
                                 = NULL;
               InitProperties[4];
DBPROP
                 rgInitPropSet[1];
DBPROPSET
```

```
DBPROPSET rgPropSets[1];
DBPROP rgProperties[1];
```

```
ULONG i, j;
HRESULT hr;
```

LONG	cNumRows = 0;
ULONG	lNumCols;
WCHAR*	pStringsBuffer;
DBBINDING*	pBindings;
ULONG	ConsumerBufferColOffset = 0;
HACCESSOR	hAccessor;
ULONG	lNumRowsRetrieved;
HROW	hRows[5];
HROW*	pRows = &hRows[0];
char*	pBuffer;

void main() {

```
//The command to execute.
WCHAR* wCmdString
= OLESTR(" SELECT title_id, title FROM titles ");
```

// Initialize and establish a connection to the data source. InitializeAndEstablishConnection();

```
cout << "pIDBCreateSession->CreateSession failed.\n";
}
```

```
//Access the ICommandText interface.
if(FAILED(pIDBCreateCommand->CreateCommand(
                  NULL,
                  IID ICommandText,
                  (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
}
//Set DBPROP IRowsetLocate
if(FAILED(pICommandText->QueryInterface(
                  IID_ICommandProperties,
                  (void **) &pICommandProperties )))
{
  cout << "Failed to obtain ICommandProperties interface.\n";
}
/*
Set DBPROP IRowsetLocate to VARIANT TRUE to
get the IRowsetLocate interface.
*/
VariantInit(&rgProperties[0].vValue);
rgPropSets[0].guidPropertySet = DBPROPSET_ROWSET;
rgPropSets[0].cProperties
                           = 1:
rgPropSets[0].rgProperties = rgProperties;
```

//Set properties in the property group (DBPROPSET\_ROWSET)
rgPropSets[0].rgProperties[0].dwPropertyID = DBPROP\_IRowsetL
rgPropSets[0].rgProperties[0].dwOptions = DBPROPOPTIONS\_]

```
rgPropSets[0].rgProperties[0].colid = DB_NULLID;
rgPropSets[0].rgProperties[0].vValue.vt = VT_BOOL;
rgPropSets[0].rgProperties[0].vValue.boolVal = VARIANT_TRUE;
```

```
pICommandProperties->Release();
```

```
cout << "Failed to set command text.\n";
}</pre>
```

```
//Execute the command.
if(FAILED(hr = pICommandText->Execute(
```

```
NULL,
                     IID_IRowset,
                     NULL,
                     &cNumRows,
                     (IUnknown **) &pIRowset)))
  {
    cout << "Failed to execute command.\n";
  }
  ProcessResultSet();
  pIRowset->Release();
  //Free up memory.
  pICommandText->Release();
  pIDBCreateCommand->Release();
  pIDBCreateSession->Release();
  if(FAILED(pIDBInitialize->Uninitialize()))
  {
    /*
    Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.
    */
    cout << "Problem uninitializing.\n";</pre>
  } //if.
  pIDBInitialize->Release();
  //Release COM library.
  CoUninitialize();
};
//_____
                   _____
void InitializeAndEstablishConnection()
```

{

```
//Initialize the COM library.
CoInitialize(NULL);
```

```
//Initialize the property values that are the same for each property.
for(i = 0; i < 5; i++) {
    VariantInit(&InitProperties[i].vValue);
    InitProperties[i].dwOptions = DBPROPOPTIONS_REQUIRED;
    InitProperties[i].colid = DB_NULLID;
}</pre>
```

```
//Server name.
```

```
//Database.
```

```
InitProperties[1].dwPropertyID = DBPROP_INIT_CATALOG;
InitProperties[1].vValue.vt = VT_BSTR;
InitProperties[1].vValue.bstrVal = SysAllocString((LPOLESTR)L"p
```

```
//Login.
InitProperties[2].dwPropertyID = DBPROP_AUTH_USERID;
InitProperties[2].vValue.vt = VT_BSTR;
InitProperties[2].vValue.bstrVal = SysAllocString((LPOLESTR)L")
```

```
//Password.
InitProperties[3].dwPropertyID = DBPROP_AUTH_PASSWORD;
InitProperties[3].vValue.vt = VT_BSTR;
InitProperties[3].vValue.bstrVal = SysAllocString((LPOLESTR)"page)
```

```
//Construct the PropertySet array.
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0].rgProperties = InitProperties;
```

```
hr = pIDBProperties->SetProperties(1, rgInitPropSet);
if(FAILED(hr))
{
    cout << "Failed to set initialization properties.\n";
}</pre>
```

```
pIDBProperties->Release();
```

```
pIRowset->QueryInterface(
IID_IColumnsInfo,
(void **)&pIColumnsInfo);
```

```
pIColumnsInfo->GetColumnInfo(
&lNumCols,
&pDBColumnInfo,
&pStringsBuffer);
```

```
//Create a DBBINDING array.
pBindings = new DBBINDING[lNumCols];
```

```
//Using the ColumnInfo strucuture, fill out the pBindings array.
for(j=0; j<lNumCols; j++) {</pre>
  pBindings[j].iOrdinal = j;
  pBindings[j].obValue = ConsumerBufferColOffset;
  pBindings[j].pTypeInfo = NULL;
  pBindings[j].pObject = NULL;
  pBindings[j].pBindExt = NULL;
  pBindings[j].dwPart = DBPART_VALUE;
  pBindings[j].dwMemOwner = DBMEMOWNER_CLIENTOWN
  pBindings[j].eParamIO = DBPARAMIO_NOTPARAM;
  pBindings[j].cbMaxLen = pDBColumnInfo[j].ulColumnSize + 1
                // + 1 for null terminator
  pBindings[j].dwFlags = 0;
  pBindings[j].wType = pDBColumnInfo[j].wType;
  pBindings[j].bPrecision = pDBColumnInfo[j].bPrecision;
  pBindings[j].bScale = pDBColumnInfo[j].bScale;
  //Recalculate the next buffer offset.
  ConsumerBufferColOffset = ConsumerBufferColOffset +
                  pDBColumnInfo[j].ulColumnSize;
```

```
};
  /*
  Indicate that the first field is used as a bookmark by setting
  dwFlags to DBCOLUMNFLAGS_ISBOOKMARK.
  */
  pBindings[0].dwFlags = DBCOLUMNFLAGS_ISBOOKMARK;
//Get IAccessor interface.
hr = pIRowset->QueryInterface(
             IID IAccessor,
             (void **)&pIAccessor);
if (FAILED(hr))
{
  printf("Failed to get IAccessor interface.\n");
}
//Create accessor.
hr = pIAccessor->CreateAccessor(
           DBACCESSOR_ROWDATA,
           lNumCols,
           pBindings,
           0,
           &hAccessor,
           NULL);
if(FAILED(hr))
{
  printf("Failed to create an accessor.\n");
}
HRESULT hr = pIRowset->QueryInterface(
                IID IRowsetLocate,
                (void **) &pIRowsetLocate);
if (FAILED(hr))
{
```

```
printf("Failed to get IRowsetLocate interface.\n");
}
  hr = pIRowsetLocate->GetRowsAt(
                  0,
                  NULL,
                  sizeof(int),
                  (BYTE *) & iBookmark,
                  0,
                   1,
                   &lNumRowsRetrieved,
                  &pRows);
  if (FAILED(hr))
  {
    printf("Calling the GetRowsAt method failed.\n");
//Create buffer and retrieve data.
pBuffer = new char[ConsumerBufferColOffset];
memset(pBuffer, 0, ConsumerBufferColOffset);
hr = pIRowset->GetData(hRows[0], hAccessor, pBuffer);
if (FAILED(hr))
{
  printf("Failed GetDataCall.\n");
}
printf("%d\t%s%s\n", &pBuffer[pBindings[0].obValue],
            &pBuffer[pBindings[1].obValue],
           &pBuffer[pBindings[2].obValue]);
pIRowset->ReleaseRows(lNumRowsRetrieved,
              hRows,
              NULL,
```

### NULL, NULL);

//Free all allocated memory. delete [] pBuffer; pIAccessor->ReleaseAccessor(hAccessor, NULL); pIAccessor->Release(); delete [] pBindings;

} //ProcessResultSet.

How To

## Fetching Columns Using IRow::GetColumns (or IRow::Open) and ISequentialStream

Large data can be bound or retrieved using the **IsequentialStream** interface. For bound columns, the status flag indicates if the data is truncated by setting **DBSTATUS\_S\_TRUNCATED**.

### To fetch columns using IRow::GetColumns (or IRow::Open) and ISequentialStream

- 1. Establish a connection to the data source.
- Execute the command (in this example, IcommandExecute::Execute() is called with IID\_IRow).
- 3. The column data can be fetched using **IRow::Open()** or **IRow::GetColumns()**.
  - a. IRow::Open() can be used to open an IsequentialStream on the row. Specify DBGUID\_STREAM to indicate that the column contains a stream of binary data (IStream or ISequentialStream can then be used to read the data from the column).
  - b. If **IRow::GetColumns()** is used, then the pData element of DBCOLUMNACCESS structure is set to point to a stream object.
- 4. **IsequentialStream::Read()** is used repeatedly to read the specified number of bytes into the consumer buffer.

Here is the sample table used by the application:

use pubs go

```
if exists (select name from sysobjects where name = 'MyTable')
  drop table MyTable
go
create table MyTable
(
  col1 int,
  col2 varchar(50),
  col3 char(50),
  col4 datetime,
  col5 float,
  col6 money,
  col7 sql_variant,
  col8 binary(50),
  col9 text,
  col10 image
)
go
/* Enter data */
insert into MyTable
values
(
  10,
  'abcdefghijklmnopqrstuvwxyz',
  'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
  '11/1/1999 11:52 AM',
  3.14,
  99.95,
  convert(nchar(50), N'AbCdEfGhIjKlMnOpQrStUvWxYz'),
  0x123456789,
  replicate('AAAABBBBB', 500),
  replicate(0x123456789, 500)
```

) go

Here is the sample code:

/\*

Example shows how to fetch a single row using IRow. In this examp one column at a time is retrieved from the row. This example illustra the use of IRow::Open() as well as IRow::GetColumns(). To read the column data, the example uses ISequentialStream::Read.

\*/

#define DBINITCONSTANTS #define INITGUID

#include <stdio.h>
#include <windows.h>
#include <iostream.h>
#include <oledb.h>
#include <sqloledb.h>

```
//constants
const int kMaxBuff = 50;
```

```
void InitializeAndEstablishConnection();
HRESULT GetColumnSize(IRow* pUnkRow, ULONG iCol);
ULONG PrintData(ULONG iCols, ULONG iStart, DBCOLUMNINE
DBCOLUMNACCESS* prgColumns);
HRESULT GetColumns(IRow* pUnkRow, ULONG iStart, ULONG iF
HRESULT GetSequentialColumn(IRow* pUnkRow, ULONG iCol, B())
```

IDBInitialize\* pIDBInitialize = NULL; IDBProperties\* pIDBProperties = NULL; IDBCreateSession\* pIDBCreateSession = NULL; IDBCreateCommand\* pIDBCreateCommand = NULL; = NULL; pICommandText ICommandText\* IRow \* pIRow = NULL: IColumnsInfo\* pIColumnsInfo = NULL: DBCOLUMNINFO\* pDBColumnInfo = NULL; pIAccessor IAccessor\* = NULL: InitProperties[4]; DBPROP DBPROPSET rgInitPropSet[1]; ULONG i, j; **HRESULT** hr; cNumRows = 0;LONG lNumCols; ULONG pStringsBuffer; WCHAR\* pBindings; DBBINDING\* ConsumerBufColOffset = 0; ULONG HACCESSOR hAccessor; lNumRowsRetrieved: ULONG hRows[10]; HROW pRows = &hRows[0]; HROW\* BYTE\* pBuffer; void main() { iidx = 0;ULONG WCHAR\* wCmdString = OLESTR(" SELECT \* FROM MyTable "); // Call a function to initialize and establish connection. InitializeAndEstablishConnection();

```
(void**) &pIDBCreateSession)))
{
  cout << "Failed to obtain IDBCreateSession interface.\n";</pre>
ł
if(FAILED(pIDBCreateSession->CreateSession(
                  NULL,
                  IID_IDBCreateCommand,
                  (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
}
//Access the ICommandText interface.
if(FAILED(pIDBCreateCommand->CreateCommand(
                  NULL,
                  IID_ICommandText,
                  (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
}
//Use SetCommandText() to specify the command text.
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
  cout << "Failed to set command text.\n";
}
//Execute the command.
if(FAILED(hr = pICommandText->Execute(NULL,
                  IID IRow,
                  NULL,
                  &cNumRows,
```

### (IUnknown \*\*) &pIRow)))

```
{
    cout << "Failed to execute command.\n";
}</pre>
```

//Get columns (one at a time) using ISequentialStream and Open

```
for(iidx = 1; iidx <= 10; iidx++)
    //the 3rd parameter is by default TRUE indicating use of ISequent
    //and Open.
    hr = GetSequentialColumn(pIRow, iidx);
//Release the Row object.
pIRow->Release();
```

```
/*
```

Now get columns (one at a time) using ISequentialStream and GetColumns. The 3rd parameter is by default TRUE indicating use of ISequentialStream and GetColumns. \*/ for(iidx = 1; iidx <= 10; iidx++)

```
hr = GetSequentialColumn(pIRow, iidx, FALSE);
```

```
//Free up memory.
  pICommandText->Release();
  pIDBCreateCommand->Release();
  pIDBCreateSession->Release();
  if(FAILED(pIDBInitialize->Uninitialize()))
  {
    /*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.
    cout << "Problem uninitializing.\n"; */</pre>
  } //endif.
  pIDBInitialize->Release();
  //Release the COM library.
  CoUninitialize();
};
//-----
HRESULT GetSequentialColumn(IRow* pUnkRow, ULONG iCol, B(
{
  HRESULT hr = NOERROR;
  ULONG cbRead = 0:
  ULONG cbTotal = 0;
  ULONG cColumns = 0;
  ULONG cReads = 0;
  ISequentialStream* pIStream = NULL;
  WCHAR* pBuffer[kMaxBuff];//50 chars read by ISequentialStream
  DBCOLUMNINFO* prgInfo;
  OLECHAR* pColNames;
  IColumnsInfo* pIColumnsInfo;
  DBID columnid:
  DBCOLUMNACCESS column;
```

```
wprintf(TEXT("[RETRIEVING COLUMN %d SEQUENTIALLY])
```

```
//Get column information (basically get column id)
hr = pUnkRow->QueryInterface(IID_IColumnsInfo,
             (void**) &pIColumnsInfo);
if(FAILED(hr))
  goto CLEANUP;
hr = pIColumnsInfo->GetColumnInfo(&cColumns, &prgInfo, &pCo
if (FAILED(hr))
  goto CLEANUP;
//Get Column ID
columnid = (prgInfo + (iCol - 1))->columnid;
if (fOpen) //Get columns using Open and ISequentialStream.
{
  wprintf(TEXT("[RETRIEVING COLUMNS USING "));
  wprintf(TEXT(" ISequentialSteam and Open]\n"));
  //Open sequential stream
  hr = pUnkRow->Open(NULL,
             &columnid,
             DBGUID STREAM,
             0,
             IID_ISequentialStream,
             (LPUNKNOWN *)&pIStream);
  if (FAILED(hr))
  {
    wprintf(TEXT("Unable to get ISequentialStream interface.\n"))
    goto CLEANUP;
  }
}
```

```
else //Get Columns using GetColumns and ISequentialStream.
```

```
{
```

```
IUnknown* pUnkStream = NULL;
  ZeroMemory(&column, sizeof(column));
  column.columnid = prgInfo[iCol - 1].columnid;
  column.wType = DBTYPE_IUNKNOWN;
  column.pData = (LPVOID*) &pUnkStream;
  hr = pUnkRow->GetColumns(1, &column);
  if (FAILED(hr))
  {
    wprintf(TEXT("Error executing IRow::GetColumns.\n"));
    goto CLEANUP;
  }
  hr = pUnkStream->QueryInterface(IID_ISequentialStream,
                   (LPVOID*) &pIStream);
  if (FAILED(hr))
  {
    wprintf(TEXT("Unable to get ISequentialStream interface "));
    wprintf(TEXT("via IRow::GetColumns.\n"));
    goto CLEANUP;
  }
  pUnkStream->Release();
ZeroMemory(pBuffer, kMaxBuff * sizeof(WCHAR));
```

```
//Read 50 chars at a time until no more data.
do
{
  hr = pIStream->Read(pBuffer, kMaxBuff, &cbRead);
```

```
if(FAILED(hr))
{
    wprintf(TEXT("Error reading data.\n"));
    goto CLEANUP;
}
cbTotal = cbTotal + cbRead;
//Print the data
wprintf(TEXT("READ #%d: %-*S\n"), ++cReads, kMaxBuff, pE
} while(cbRead > 0);
```

```
wprintf(TEXT("[READ %d bytes for column %d.\n"), cbTotal, iCol
```

```
CLEANUP:
if(pIColumnsInfo)
pIColumnsInfo->Release();
```

```
if(pIStream)
    pIStream->Release();
```

```
return hr;
```

```
//-----
```

```
BOOL InitColumn(DBCOLUMNACCESS* pCol, DBCOLUMNINF(
{
```

```
/*
```

If text or image column is being read, in which case the max possibl length of a value is the column is hugh, we will limit that size to 512 bytes (for illustration purposes). \*/

```
ULONG ulSize=
(pInfo->ulColumnSize < 0x7fffffff) ? pInfo->ulColumnSize : 512
```

```
//Verify dta buffer is large enough.
if(pCol->cbMaxLen < (ulSize + 1))
ł
  if(pCol->pData)
  {
    delete [] pCol->pData;
    pCol->pData = NULL;
  }
  //Allocate data buffer
  pCol->pData = new WCHAR[ulSize + 1];
  //set the max length of caller-initialized memory.
  pCol->cbMaxLen = sizeof(WCHAR) * (ulSize + 1);
  /*
  In the above 2 steps, pData is pointing to memory (it is not NULL
  and cbMaxLen has a value (not 0), so next call to IRow->GetData
  will read the data from the column.
  */
}
```

```
//Clear memory buffer
ZeroMemory((void*) pCol->pData, pCol->cbMaxLen);
```

```
//Set properties.
pCol->wType = DBTYPE_WSTR;
pCol->columnid = pInfo->columnid;
pCol->cbDataLen = 0;
pCol->dwStatus = 0;
pCol->dwReserved = 0;
pCol->bPrecision = 0;
pCol->bScale = 0;
```

```
return TRUE;
```

```
//-----
HRESULT GetColumns(IRow* pUnkRow, ULONG iStart, ULONG iI
```

//Start and end are same. Thus, get only one column.

```
{
```

```
HRESULT hr;
ULONG iidx; //loop counter
ULONG cColumns; //Count of columns
ULONG cUserCols; //Count of user columns
DBCOLUMNINFO* prgInfo; //Column of info. array
OLECHAR* pColNames; //Array of column names
DBCOLUMNACCESS* prgColumns; //Ptr to column access structu
DBCOLUMNINFO* pCurrInfo;
DBCOLUMNACCESS* pCurrCol;
```

IColumnsInfo\* pIColumnsInfo = NULL;

```
//Initialize
cColumns = 0;
prgInfo = NULL;
pColNames = NULL;
prgColumns = NULL;
```

```
printf("Retrieving data\n");
```

```
//Get column info to build column access array
hr=pUnkRow->QueryInterface(IID_IColumnsInfo, (void**)&pICol
if(FAILED(hr))
goto CLEANUP;
hr=pIColumnsInfo->GetColumnInfo(&cColumns, &prgInfo, &pCol
if(FAILED(hr))
```

goto CLEANUP;

```
printf("In GetColumns(), Columns= %d\n", cColumns);
```

```
/*
Determine no. of columns to retrieve.
Since iEnd and iStart is same, this is redundent step.
cUserCols will always be 1.
*/
cUserCols = iEnd - iStart + 1;
//Walk list of columns and setup a DBCOLUMNACCESS structure
prgColumns= new DBCOLUMNACCESS[cUserCols]; //cUserCols
ZeroMemory((void*) prgColumns, sizeof(DBCOLUMNACCESS) *
```

```
for(iidx=0; iidx < cUserCols; iidx++)</pre>
ł
  pCurrInfo = prgInfo + iidx + iStart - 1;
  pCurrCol = prgColumns + iidx;
  //Here the values of pData and cbMaxLen elements of
  //DBCOLUMNACCESS elements is set. Thus IRow->GetColum
  //will return actual data.
  if(InitColumn(pCurrCol, pCurrInfo) == FALSE)
    goto CLEANUP;
}
hr = pUnkRow->GetColumns(cUserCols, prgColumns); //cUserCols
if(FAILED(hr))
{
  printf("Error occured\n");
}
//Show data.
PrintData(cUserCols, iStart, prgInfo, prgColumns);
```

```
CLEANUP:

if(pIColumnsInfo)

pIColumnsInfo->Release();

if(prgColumns)

delete [] prgColumns;
```

return hr;

```
}
//-----
```

/\*

This function returns the actual width of the data in the column (not the columnwidth in DBCOLUMNFO structure which is the width column)

\*/

```
HRESULT GetColumnSize(IRow* pUnkRow, ULONG iCol) {

HRESULT hr = NOERROR;

ULONG iidx = 0; //Loop counter

ULONG cColumns = 0; //Count the columns

DBCOLUMNINFO* prgInfo; //Column info array

OLECHAR* pColNames;

DBCOLUMNACCESS column;

DBCOLUMNINFO* pCurrInfo;

IColumnsInfo* pIColumnsInfo = NULL;
```

//Initialize
prgInfo = NULL;
pColNames = NULL;

printf("Checking column size\n");

//Get column info to build column access array

```
hr=pUnkRow->QueryInterface(IID_IColumnsInfo, (void**) &pICo
if (FAILED(hr))
goto CLEANUP;
```

```
hr=pIColumnsInfo->GetColumnInfo(&cColumns, &prgInfo, &pCol
if (FAILED(hr))
```

```
goto CLEANUP;
```

```
printf("Value of cColumns is %d\n", cColumns);
```

/\*

Setup a DBCOLUMNACCESS structure: Here pData is set to NUL cbMaxLen is set to 0. Thus IRow->GetColumns() returns only the a column length in cbDataLen member of DBCOLUMNACCESS stru case you can call IRow->GetColumns() again for the same column t retrieve actual data in the second call.

\*/

ZeroMemory((void\*) &column, sizeof(DBCOLUMNACCESS)); column.pData=NULL;

```
pCurrInfo = prgInfo + iCol - 1;
//Get the column id in DBCOLUMNACCESS structure.
//It is then used in GetColumn().
column.columnid = pCurrInfo->columnid;
```

```
printf("column.columnid value is %d\n", column.columnid);
//We know which column to get.
//The column.columnid gives the column number.
hr = pUnkRow->GetColumns(1, &column);
if (FAILED(hr))
{
    printf("Errors occured\n");
}
//Show data
```

```
PrintData(1, iCol, prgInfo, &column);
CLEANUP:
 if (pIColumnsInfo)
   pIColumnsInfo->Release();
 return hr;
}
//-----
BOOL GetStatus(DWORD dwStatus, WCHAR* pwszStatus)
{
 switch (dwStatus)
```

```
{
case DBSTATUS S OK:
  wcscpy(pwszStatus, TEXT("DBSTATUS_S_OK"));
```

```
break;
```

```
case DBSTATUS_E_UNAVAILABLE:
```

```
wcscpy(pwszStatus, TEXT("DBSTATUS_E_UNAVAILABLE"))
break;
```

```
case DBSTATUS S TRUNCATED:
```

```
wcscpy(pwszStatus, TEXT("DBSTATUS_S_TRUNCATED"));
break;
```

```
}
return TRUE;
```

```
}
//_____
```

```
ULONG PrintData(ULONG iCols,
```

```
ULONG iStart,
DBCOLUMNINFO* prgInfo,
DBCOLUMNACCESS* prgColumns)
```

```
{
```

```
WCHAR wszStatus[255];
```

#### DBCOLUMNINFO\* pCurrInfo; DBCOLUMNACCESS\* pCurrCol;

```
printf("No. Name
                     Status
                             Length Max Data\n");
  for(ULONG iidx=0; iidx < iCols; iidx++)
  {
    pCurrInfo=prgInfo + iidx + iStart - 1;
    pCurrCol=prgColumns+iidx;
    GetStatus(pCurrCol->dwStatus, wszStatus);
    //was the data successfully retrieved?
    wprintf(TEXT("%-3d %-*s %-20s %-3d %-3d %-20s\n"),
      iStart+iidx,
      10,
      pCurrInfo->pwszName,
      wszStatus,
      pCurrCol->cbDataLen,
      pCurrCol->cbMaxLen,
      (WCHAR*) pCurrCol->pData);
  }
  wprintf(TEXT("\n"));
  return iidx;
}
//_____
void InitializeAndEstablishConnection()
{
  //Initialize the COM library.
  CoInitialize(NULL);
```

//Obtain access to the SQLOLEDB provider.

//Database.

```
InitProperties[1].dwPropertyID = DBPROP_INIT_CATALOG;
InitProperties[1].vValue.vt = VT_BSTR;
InitProperties[1].vValue.bstrVal= SysAllocString(L"pubs");
InitProperties[1].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[1].colid = DB_NULLID;
```

```
//Username (login).
```

InitProperties[2].dwPropertyID = DBPROP\_AUTH\_USERID; InitProperties[2].vValue.vt = VT\_BSTR; InitProperties[2].vValue.bstrVal= SysAllocString(L"sa"); InitProperties[2].dwOptions = DBPROPOPTIONS\_REQUIRED; InitProperties[2].colid = DB\_NULLID;

//Password.

```
InitProperties[3].dwPropertyID = DBPROP_AUTH_PASSWORD;
InitProperties[3].vValue.vt = VT_BSTR;
InitProperties[3].vValue.bstrVal= SysAllocString(L''');
InitProperties[3].dwOptions = DBPROPOPTIONS_REQUIRED;
InitProperties[3].colid = DB_NULLID;
```

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0].rgProperties = InitProperties;
```

```
if(FAILED(hr))
```

```
{
    cout << "Failed to set initialization properties.\n";
    //end if
    pIDBProperties->Release();
    //Now establish the connection to the data source.
    if(FAILED(pIDBInitialize->Initialize()))
    {
        cout << "Problem establishing connection to the data source.\n";
    }
} //end of InitializeAndEstablishConnection.
//--------</pre>
```

How To

# Fetching Columns Using IRow::GetColumns (OLE DB)

The **IRow** interface allows direct access to columns of a single row in the result set. Thus, **IRow** is an efficient way to retrieve columns from a result set with one row.

#### To fetch columns using IRow::GetColumns

- 1. Establish a connection to the data source.
- Execute the command (in the following example, IcommandExecute::Execute() is called with IID\_IRow).
- 3. Execute **IRow::GetColumns()** to fetch one or more columns in the resulting row. If you want to find the actual column size before fetching data, set the pData in DBCOLUMNACCESS to NULL. The call to IRow::GetColumns() will return only the column width. Another call the IRow::GetColumns() will fetch the data.
- 4. Execute **IRow::GetColumns()** until all the columns you need are accessed. The columns must be accessed in sequence.

This example shows how to fetch a single row using **IRow**. It also illustrates two ways to access columns in the row:

- Fetching columns in groups, and
- Obtaining the column width first and then fetch the column data.

Here is the sample table used by the application:

use pubs

go

```
if exists (select name from sysobjects where name = 'MyTable')
  drop table MyTable
go
create table MyTable
(
  col1 int,
  col2 varchar(50),
  col3 char(50),
  col4 datetime,
  col5 float,
  col6 money,
  col7 sql_variant,
  col8 binary(50),
  col9 text,
  col10 image
)
go
insert into MyTable
values
(
  10.
  'abcdefghijklmnopqrstuvwxyz',
  'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
  '11/1/1999 11:52 AM',
  3.14,
  99.95,
  convert(nchar(50), N'AbCdEfGhIjKlMnOpQrStUvWxYz'),
  0x123456789,
  replicate('AAAABBBBB', 500),
  replicate(0x123456789, 500)
)
go
```

Here is the sample code:

/\*

This example shows how to fetch a single row using IRow. In this example one column at a time is retrieved from the row. The exampl shows 2 things

1) How to fetch a group of columns (in sequence)

2) How to access a column twice - the first time the actual column width is obtained and then later the actual data is accessed. In the DBCOLUMNACCESS structure, if pData is NULL and cbMaxLen call to IRow->GetColumns() returns only the actual column length. this case IRow->GetColumns() can be called again on the same column to retrieve the actual data.

\*/

#define DBINITCONSTANTS

```
#include <stdio.h>
#include <windows.h>
#include <iostream.h>
#include <oledb.h>
#include <sqloledb.h>
```

```
    void InitializeAndEstablishConnection();
    HRESULT GetColumnSize(IRow* pUnkRow, ULONG iCol);
    ULONG PrintData(ULONG iCols, ULONG iStart, DBCOLUMNIN DBCOLUMNACCESS* prgColumns);
    HRESULT GetColumns(IRow* pUnkRow, ULONG iStart, ULONG iF
```

IDBInitialize\* pIDBInitialize = NULL; IDBProperties\* pIDBProperties = NULL; IDBCreateSession\* pIDBCreateSession = NULL; IDBCreateCommand\* pIDBCreateCommand = NULL; ICommandText\* pICommandText = NULL; IRow \* pIRow = NULL;

```
IColumnsInfo*
                pIColumnsInfo
                                 = NULL;
                      pDBColumnInfo
DBCOLUMNINFO*
                                      = NULL;
IAccessor*
               pIAccessor
                             = NULL;
DBPROP
                InitProperties[4];
                  rgInitPropSet[1];
DBPROPSET
ULONG
               i, j;
HRESULT
                hr;
LONG
              cNumRows = 0;
               lNumCols;
ULONG
WCHAR*
                pStringsBuffer;
DBBINDING*
                  pBindings;
                ConsumerBufColOffset = 0;
ULONG
HACCESSOR
                  hAccessor;
               lNumRowsRetrieved;
ULONG
               hRows[10];
HROW
HROW*
                pRows = &hRows[0];
BYTE*
               pBuffer;
void main()
{
  ULONG
            iidx = 0;
  WCHAR* wCmdString
    = OLESTR(" SELECT * FROM MyTable ");
 // Call a function to initialize and establish connection.
  InitializeAndEstablishConnection();
  //Create a session object.
```

```
cout << "Failed to obtain IDBCreateSession interface.\n";
}</pre>
```

```
if(FAILED(pIDBCreateSession->CreateSession(
                  NULL,
                  IID_IDBCreateCommand,
                  (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
}
//Access the ICommandText interface.
if(FAILED(pIDBCreateCommand->CreateCommand(
                 NULL,
                 IID_ICommandText,
                 (IUnknown**) &pICommandText)))
{
  cout << "Failed to access ICommand interface.\n";
}
//Use SetCommandText() to specify the command text.
if(FAILED(pICommandText->SetCommandText(DBGUID_DBSQI
{
  cout << "Failed to set command text.\n";
}
/*
Fetch columns 1-5 and then 6-10 and display the contents
*/
if(FAILED(hr = pICommandText->Execute(NULL,
                 IID_IRow,
                 NULL,
                 &cNumRows,
                 (IUnknown **) &pIRow)))
{
```

```
cout << "Failed to execute command.\n";
}
hr = GetColumns(pIRow, 1, 5);
hr = GetColumns(pIRow, 6, 10);
hr = pIRow->Release();
//Execute the command.
if(FAILED(hr = pICommandText->Execute(NULL,
                  IID_IRow,
                  NULL,
                  &cNumRows,
                  (IUnknown **) &pIRow)))
{
  cout << "Failed to execute command.\n";
}
//Get columns
for(iidx=1; iidx <=10; iidx++)</pre>
{
   hr=GetColumnSize(pIRow, iidx);
   hr=GetColumns(pIRow, iidx, iidx);
}
pIRow->Release();
//Free up memory.
pICommandText->Release();
pIDBCreateCommand->Release();
pIDBCreateSession->Release();
if(FAILED(pIDBInitialize->Uninitialize()))
{
```

```
/*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.
    cout << "Problem uninitializing.\n"; */</pre>
  } //endif.
  pIDBInitialize->Release();
  //Release the COM library.
  CoUninitialize();
};
//_____
BOOL InitColumn(DBCOLUMNACCESS* pCol, DBCOLUMNINF(
{
  //If text or image column is being read, in which case the max
  // possible length of a value is the column is hugh, we will limit
  //that size to 512 bytes (for illustration purposes).
  ULONG ulSize=
     (pInfo->ulColumnSize < 0x7fffffff) ? pInfo->ulColumnSize : 51
  //Verify dta buffer is large enough.
  if(pCol->cbMaxLen < (ulSize + 1))
   {
     if(pCol->pData)
      {
        delete [] pCol->pData;
        pCol->pData = NULL;
```

```
//Allocate data buffer
pCol->pData = new WCHAR[ulSize + 1];
//set the max length of caller-initialized memory.
pCol->cbMaxLen = sizeof(WCHAR) * (ulSize + 1);
/*
```

```
In the above 2 steps, pData is pointing to memory (it is not NULL) and cbMaxLen has a value (not 0), so next call to IRow->GetData() will read the data from the column. */
```

```
//Clear memory buffer
ZeroMemory((void*) pCol->pData, pCol->cbMaxLen);
```

```
//Set properties.
pCol->wType = DBTYPE_WSTR;
pCol->columnid = pInfo->columnid;
pCol->cbDataLen = 0;
pCol->dwStatus = 0;
pCol->dwReserved = 0;
pCol->bPrecision = 0;
pCol->bScale = 0;
return TRUE;
```

}

```
//-----
```

HRESULT GetColumns(IRow\* pUnkRow, ULONG iStart, ULONG iF //Start and end are same. Thus, get only one column.

```
{
```

```
HRESULT
                hr;
ULONG
              iidx;
                      //loop counter
ULONG
              cColumns; //Count of columns
              cUserCols; //Count of user columns
ULONG
DBCOLUMNINFO*
                   prgInfo; //Column of info. array
               pColNames; //Array of column names
OLECHAR*
DBCOLUMNACCESS* prgColumns; //Ptr to column access struct
DBCOLUMNINFO*
                   pCurrInfo;
DBCOLUMNACCESS* pCurrCol;
```

IColumnsInfo\* pIColumnsInfo = NULL;

```
//Initialize
cColumns = 0;
prgInfo = NULL;
pColNames = NULL;
prgColumns = NULL;
```

```
printf("Retrieving data\n");
```

```
//Get column info to build column access array
hr=pUnkRow->QueryInterface(IID_IColumnsInfo, (void**)&pICo
if(FAILED(hr))
goto CLEANUP;
hr=pIColumnsInfo->GetColumnInfo(&cColumns, &prgInfo, &pCo
if(FAILED(hr))
goto CLEANUP;
```

```
printf("In GetColumns(), Columns= %d\n", cColumns);
```

/\*

```
Determine no. of columns to retrieve.
Since iEnd and iStart is same, this is redundent step.
cUserCols will always be 1.
*/
cUserCols = iEnd - iStart + 1;
//Walk list of columns and setup a DBCOLUMNACCESS structure
prgColumns= new DBCOLUMNACCESS[cUserCols]; //cUserCols
ZeroMemory((void*) prgColumns, sizeof(DBCOLUMNACCESS)
```

```
for(iidx=0; iidx < cUserCols; iidx++)</pre>
```

```
{
    pCurrInfo = prgInfo + iidx + iStart - 1;
    pCurrCol = prgColumns + iidx;
    //Here the values of DBCOLUMNACCESS elements is set
    //(pData and cbMaxLen)Thus IRow->GetColumns() will return {
        //data.
        if(InitColumn(pCurrCol, pCurrInfo) == FALSE)
            goto CLEANUP;
    }
hr = pUnkRow->GetColumns(cUserCols, prgColumns); //cUserCol
    if(FAILED(hr))
    {
        printf("Error occured\n");
    }
```

```
//Show data.
PrintData(cUserCols, iStart, prgInfo, prgColumns);
```

```
CLEANUP:
```

```
if(pIColumnsInfo)
    pIColumnsInfo->Release();
if(prgColumns)
    delete [] prgColumns;
```

```
return hr;
```

```
}
//------
/*
```

This function returns the actual width of the data in the column (not the columnwidth in DBCOLUMNFO structure which is the width column)

\*/

HRESULT GetColumnSize(IRow\* pUnkRow, ULONG iCol) {

```
hr = NOERROR;
HRESULT
ULONG
              iidx = 0; //Loop counter
ULONG
              cColumns = 0; //Count the columns
DBCOLUMNINFO*
                   prgInfo; //Column info array
               pColNames;
OLECHAR*
DBCOLUMNACCESS
                     column:
DBCOLUMNINFO*
                   pCurrInfo;
IColumnsInfo*
              pIColumnsInfo = NULL;
```

//Initialize
prgInfo = NULL;
pColNames = NULL;

printf("Checking column size\n");

```
//Get column info to build column access array
hr=pUnkRow->QueryInterface(IID_IColumnsInfo, (void**) &pICc
if (FAILED(hr))
goto CLEANUP;
```

```
hr=pIColumnsInfo->GetColumnInfo(&cColumns, &prgInfo, &pCo
if (FAILED(hr))
   goto CLEANUP;
printf("Value of cColumns is %d\n", cColumns);
```

/\*

Setup a DBCOLUMNACCESS structure: Here pData is set to NUL cbMaxLen is set to 0. Thus IRow->GetColumns() returns only the actual column length in cbDataLen member of DBCOLUMNACCE In this case you can call IRow->GetColumns() again for the same column to retrieve actual data in the second call.

\*/

}

{

ZeroMemory((void\*) &column, sizeof(DBCOLUMNACCESS)); column.pData=NULL;

```
pCurrInfo = prgInfo + iCol - 1;
//Get the column id in DBCOLUMNACCESS structure.
//It is then used in GetColumn().
column.columnid = pCurrInfo->columnid;
```

```
printf("column.columnid value is %d\n", column.columnid);
  //We know which column to get. The column.columnid gives the co
  hr = pUnkRow->GetColumns(1, &column);
  if (FAILED(hr))
  {
     printf("Errors occured\n");
  }
  //Show data
  PrintData(1, iCol, prgInfo, &column);
CLEANUP:
  if (pIColumnsInfo)
     pIColumnsInfo->Release();
  return hr;
//_____
BOOL GetStatus(DWORD dwStatus, WCHAR* pwszStatus)
  switch (dwStatus)
  case DBSTATUS S OK:
     wcscpy(pwszStatus, TEXT("DBSTATUS_S_OK"));
     break:
  case DBSTATUS E UNAVAILABLE:
```

```
wcscpy(pwszStatus, TEXT("DBSTATUS_E_UNAVAILABLE")
     break:
  case DBSTATUS S TRUNCATED:
     wcscpy(pwszStatus, TEXT("DBSTATUS_S_TRUNCATED"));
     break:
  }
  return TRUE:
}
//_____
ULONG PrintData(ULONG iCols,
          ULONG iStart,
          DBCOLUMNINFO* prgInfo,
          DBCOLUMNACCESS* prgColumns)
{
  WCHAR wszStatus[255];
  DBCOLUMNINFO* pCurrInfo;
  DBCOLUMNACCESS* pCurrCol;
  printf("No. Name Status Length Max Data\n");
  for(ULONG iidx=0; iidx < iCols; iidx++)</pre>
  {
     pCurrInfo=prgInfo + iidx + iStart - 1;
     pCurrCol=prgColumns+iidx;
     GetStatus(pCurrCol->dwStatus, wszStatus);
     //was the data successfully retrieved?
     wprintf(TEXT("%-3d %-*s %-20s %-3d %-3d %-20s\n"),
       iStart+iidx,
       10,
       pCurrInfo->pwszName,
       wszStatus,
```

```
pCurrCol->cbDataLen,
    pCurrCol->cbMaxLen,
    (WCHAR*) pCurrCol->pData);
}
wprintf(TEXT("\n"));
return iidx;
}
```

```
} //end if
```

```
/*
Initialize the property values needed
to establish the connection.
*/
for(i = 0; i < 4; i++)
VariantInit(&InitProperties[i].vValue);
```

```
//Server name.
  InitProperties[0].dwPropertyID = DBPROP_INIT_DATASOURCE
  InitProperties[0].vValue.vt
                            = VT BSTR;
  InitProperties[0].vValue.bstrVal=
               SysAllocString(L"mohanv1");
  InitProperties[0].dwOptions = DBPROPOPTIONS_REQUIRED;
                          = DB_NULLID;
  InitProperties[0].colid
//Database.
  InitProperties[1].dwPropertyID = DBPROP_INIT_CATALOG;
  InitProperties[1].vValue.vt
                            = VT BSTR;
  InitProperties[1].vValue.bstrVal= SysAllocString(L"pubs");
  InitProperties[1].dwOptions = DBPROPOPTIONS_REQUIRED;
                          = DB_NULLID;
  InitProperties[1].colid
//Username (login).
  InitProperties[2].dwPropertyID = DBPROP_AUTH_USERID;
  InitProperties[2].vValue.vt
                            = VT BSTR;
  InitProperties[2].vValue.bstrVal= SysAllocString(L"sa");
  InitProperties[2].dwOptions = DBPROPOPTIONS REQUIRED;
  InitProperties[2].colid
                          = DB NULLID;
//Password.
  InitProperties[3].dwPropertyID = DBPROP AUTH PASSWORD;
  InitProperties[3].vValue.vt
                            = VT_BSTR;
  InitProperties[3].vValue.bstrVal= SysAllocString(L''');
```

InitProperties[3].dwOptions = DBPROPOPTIONS\_REQUIRED; InitProperties[3].colid = DB\_NULLID;

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method.

```
*/
  rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
  rgInitPropSet[0].cProperties = 4;
  rgInitPropSet[0].rgProperties = InitProperties;
  //Set initialization properties.
  hr = pIDBInitialize->QueryInterface(IID_IDBProperties,
                    (void **)&pIDBProperties);
  if(FAILED(hr))
  {
    cout << "Failed to get IDBProperties interface.\n";
  }
  hr = pIDBProperties->SetProperties(1, rgInitPropSet);
  if(FAILED(hr))
  ł
    cout << "Failed to set initialization properties.\n";</pre>
  } //end if
  pIDBProperties->Release();
  //Now establish the connection to the data source.
  if(FAILED(pIDBInitialize->Initialize()))
  {
    cout << "Problem establishing connection to the data source.\n";
  }
} //end of InitializeAndEstablishConnection.
//_____
```

How To

## Setting XML as a Command Using ICommandStream and Retrieving the Results as an XML Document

The **ICommandStream** interface can be used to set XML documents as a command, and the results can be retrieved as an XML document.

## **Executing Templates with XPath Queries**

The following XML template consisting of an XPath query is specified as a command using **ICommandStream**:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
<sql:xpath-query mapping-schema="Schema.xml">Employees</sql:xq
</ROOT>
```

The XPath query in the template is executed against the following mapping schema:

```
<?xml version="1.0" ?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="ui
<ElementType name="Employees" >
<AttributeType name="EmployeeID" />
<AttributeType name="FirstName" />
<AttributeType name="LastName" />
<attribute type="EmployeeID" />
<attribute type="FirstName" />
<attribute type="FirstName" />
<Attribute type="LastName" />
</ElementType>
</Schema>
```

The query returns all of the employee elements. With default mapping, the <Employees> element maps to the **Employees** table in the **Northwind** database.

## To set XML as a command and retrieving result as an XML document

1. Initialize and establish a connection to the database.

- 2. Obtain **ICommandStream** interface on **ICommand**.
- 3. Set the necessary command properties. In this example, provider specific property **SSPROP\_STREAM\_BASEPATH** is set to the directory where the mapping schema and the template files are stored.
- 4. Use **ICommandStream::SetCommandStream** to specify the command stream. In this example, the XML template being executed is read from a file. This is useful when you want to execute large XML templates.
- 5. Execute the XML command using **ICommand::Execute**, requesting IID\_ISequentialStream interface ID.
- 6. Process the result. In this example, the XML read from the stream is displayed on the screen.

void InitializeAndEstablishConnection(); void SetCommandProperties(); void ProcessResultSet();

#define UNICODE
#define \_UNICODE
#define DBINITCONSTANTS
#define INITGUID

#include <stdio.h>
#include <tchar.h>
#include <stddef.h>
#include <windows.h>
#include <iostream.h>
#include <oledb.h>

```
#include <SQLOLEDB.h>
```

```
class CSequentialStream : public ISequentialStream
{
    private:
```

ULONG	m_cRef;	// I	reference count
HANDLE	m_hFile;	// ha	ndle to the file
LPWSTR	m_wszFileName;		// the file name

public:

CSequentialStream( LPWSTR ); virtual ~CSequentialStream();

// IUnknown Methods
STDMETHODIMP\_(ULONG) AddRef();
STDMETHODIMP\_(ULONG) Release();
STDMETHODIMP QueryInterface( REFIID, LPVOID\* );

```
// ISequentialStream Methods
STDMETHODIMP Read(
    /* [out] */ void __RPC_FAR*,
    /* [in] */ ULONG,
    /* [out] */ ULONG __RPC_FAR* );
```

```
STDMETHODIMP Write(
```

/\* [in] \*/ const void \_\_\_RPC\_FAR\*,

```
/* [in] */ ULONG,
```

};

```
IDBInitialize* pIDBInitialize = NULL;
IDBProperties* pIDBProperties = NULL;
IDBCreateSession* pIDBCreateSession = NULL;
```

IDBCreateCommand\* pIDBCreateCommand = NULL; pICommand = NULL; ICommand\* pICommandStream = NULL; ICommandStream\* pICommandProperties = NULL; ICommandProperties\* = NULL; IColumnsInfo\* pIColumnsInfo ISequentialStream\* pIXMLOutput = NULL; DBCOLUMNINFO\* pDBColumnInfo = NULL; IAccessor\* pIAccessor = NULL; InitProperties[4]; DBPROP **DBPROPSET** rgInitPropSet[1]; **ULONG** i, j; hr; HRESULT cNumRows = 0; LONG lNumCols; ULONG pStringsBuffer; WCHAR\* pBindings; DBBINDING\* ConsumerBufColOffset = 0; **ULONG** HACCESSOR hAccessor: lNumRowsRetrieved; ULONG HROW hRows[10]; pRows = &hRows[0]; HROW\* BYTE\* pBuffer; XMLInput( L"Query.xml" ); CSequentialStream CSequentialStream::CSequentialStream ( LPWSTR wszFileName ) m\_cRef( 0 ), m\_hFile( NULL ), m\_wszFileName( NULL ) {

```
// The constructor AddRefs.
  AddRef();
 // Allocate memory for the file name.
 m_wszFileName = (LPWSTR) CoTaskMemAlloc( ( wcslen( wszFile
 // Copy the file name.
 wcscpy( m_wszFileName, wszFileName );
}
CSequentialStream::~CSequentialStream
(
)
{
 // Free any allocated memory.
 if( m_wszFileName )
   CoTaskMemFree( m_wszFileName );
 // Close the file.
 if( m_hFile )
   CloseHandle( m_hFile );
}
ULONG
CSequentialStream::AddRef
(
)
{
  return ++m_cRef;
```

}

```
ULONG
CSequentialStream::Release()
{
  if(--m_cRef)
    return m_cRef;
  delete this;
  return 0;
}
HRESULT
CSequentialStream::QueryInterface
(
 REFIID riid,
 void** ppv
)
{
  *ppv = NULL;
  if (riid == IID_IUnknown)
    *ppv = this;
  if (riid == IID_ISequentialStream)
    *ppv = this;
  if(*ppv)
  {
    ((IUnknown*)*ppv)->AddRef();
    return S_OK;
  }
  return E_NOINTERFACE;
}
```

```
HRESULT
CSequentialStream::Read
(
 void *pv,
 ULONG cb,
 ULONG* pcbRead
)
{
 ULONG cBytesRead = 0;
  // Parameter checking.
  if(pcbRead)
    *pcbRead = 0;
  if(!pv)
    return STG_E_INVALIDPOINTER;
  if(cb == 0)
    return S_OK;
 // Do we need to open the file?
 if( m_hFile == NULL )
 {
   // Open the file.
   m_hFile = CreateFile( m_wszFileName, GENERIC_READ, 0, NU
   // If the file failed to open, return E_FAIL.
   if( m_hFile == INVALID_HANDLE_VALUE )
     return E_FAIL;
 }
```

```
// Clear the buffer.
```

ZeroMemory( pv, cb );

```
// Read cb bytes from the stream.
 if( !ReadFile( m_hFile, pv, cb, &cBytesRead, NULL ) )
   return E_FAIL;
 // Inform the user of how many bytes to read.
 if( NULL != pcbRead )
   *pcbRead = cBytesRead;
  if(cb != cBytesRead)
    return S FALSE;
  return S_OK;
}
HRESULT
CSequentialStream::Write
(
 const void *pv,
 ULONG cb,
 ULONG* pcbWritten
)
{
 // For this example, only a read-only stream is needed.
 return STG_E_CANTSAVE;
}
void main()
{
 // Call a function to initialize and establish a connection.
  InitializeAndEstablishConnection();
```

```
// Create a session object.
if(FAILED(pIDBInitialize->QueryInterface(
                IID IDBCreateSession,
                (void**) &pIDBCreateSession)))
{
  cout << "Failed to obtain IDBCreateSession interface.\n";</pre>
}
if(FAILED(pIDBCreateSession->CreateSession(
                   NULL,
                   IID IDBCreateCommand,
                   (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
}
// Access the ICommand interface.
if(FAILED(pIDBCreateCommand->CreateCommand(
                   NULL,
                   IID ICommand,
                   (IUnknown**) &pICommand)))
{
  cout << "Failed to access ICommand interface.\n";
}
// Get an ICommandStream interface.
if(FAILED(pICommand->QueryInterface( IID_ICommandStream, (v
ł
 cout << "Failed to get an ICommandStream interface.\n";
}
// Get an ICommandProperties interface.
```

```
if(FAILED(pICommand->QueryInterface( IID_ICommandProperties
```

```
{
    cout << "Failed to get an ICommandProperties interface.\n";
}</pre>
```

```
// Set the command properties.
SetCommandProperties();
```

```
// Use SetCommandStream() to specify the command stream.
if(FAILED(pICommandStream->SetCommandStream(IID_ISequen
{
  cout << "Failed to set command stream.\n";
}
// Execute the command.
if(FAILED(hr = pICommand->Execute(NULL,
                  IID_ISequentialStream,
                  NULL,
                  &cNumRows,
                  (IUnknown **) &pIXMLOutput )))
{
  cout << "Failed to execute command.\n";
}
// Process the result set.
ProcessResultSet();
// Free memory.
if( pIXMLOutput )
 pIXMLOutput->Release();
pICommandProperties->Release();
pICommandStream->Release();
pICommand->Release();
```

```
pIDBCreateCommand->Release();
```

pIDBCreateSession->Release();

```
if(FAILED(pIDBInitialize->Uninitialize()))
  ł
    /*Uninitialize is not required, but it fails if an interface
    has not been released. This can be used for debugging.
    cout << "Problem uninitializing.\n"; */</pre>
  } // endif.
  pIDBInitialize->Release();
  // Release the COM library.
  CoUninitialize();
};
//_____
void InitializeAndEstablishConnection()
{
  // Initialize the COM library.
  CoInitialize(NULL);
  // Obtain access to the SQLOLEDB Provider.
  hr = CoCreateInstance(CLSID_SQLOLEDB,
               NULL,
               CLSCTX INPROC SERVER,
               IID_IDBInitialize,
               (void **) &pIDBInitialize);
  if(FAILED(hr))
  ł
    printf("Failed to get IDBInitialize interface.\n");
  } // end if
```

/\* Initialize the property values needed

```
to establish the connection.
*/
for(i = 0; i < 4; i++)
VariantInit(&InitProperties[i].vValue);</pre>
```

// Database.

InitProperties[1].dwPropertyID = DBPROP\_INIT\_CATALOG; InitProperties[1].vValue.vt = VT\_BSTR; InitProperties[1].vValue.bstrVal= SysAllocString(L"northwind"); InitProperties[1].dwOptions = DBPROPOPTIONS\_REQUIRED; InitProperties[1].colid = DB\_NULLID;

// Username (login).

InitProperties[2].dwPropertyID = DBPROP\_AUTH\_USERID; InitProperties[2].vValue.vt = VT\_BSTR; InitProperties[2].vValue.bstrVal= SysAllocString(L"Login"); InitProperties[2].dwOptions = DBPROPOPTIONS\_REQUIRED; InitProperties[2].colid = DB\_NULLID;

// Password.

- // InitProperties[3].dwPropertyID = DBPROP\_AUTH\_PASSWORE
- // InitProperties[3].vValue.vt = VT\_BSTR;
- // InitProperties[3].vValue.bstrVal= SysAllocString(L"Password");
- // InitProperties[3].dwOptions = DBPROPOPTIONS\_REQUIREE

// InitProperties[3].colid = DB\_NULLID;

```
/*
```

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
```

```
// rgInitPropSet[0].cProperties = 4;
rgInitPropSet[0].cProperties = 3;
```

```
rgInitPropSet[0].rgProperties = InitProperties;
```

```
{
```

```
cout << "Failed to set initialization properties.\n";
} // end if</pre>
```

```
pIDBProperties->Release();
```

```
// Establish the connection to the data source.
if(FAILED(pIDBInitialize->Initialize()))
{
    cout << "Problem establishing connection to the data source.\n";
}</pre>
```

```
InitProperties[0].colid = DB_NULLID;
```

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (InitProperties) to the SetProperties method. \*/

```
rgInitPropSet[0].guidPropertySet = DBPROPSET_SQLSERVERST
rgInitPropSet[0].cProperties = 1;
rgInitPropSet[0].rgProperties = InitProperties;
```

// Set initialization properties. hr = pICommandProperties->SetProperties(1, rgInitPropSet); if(FAILED(hr)) { cout << "Failed to set command properties.\n"; } // end if

```
} // End of InitializeAndEstablishConnection.
```

```
//-----
// Retrieve and display data resulting from a query.
void ProcessResultSet()
{
    CHAR szBuf[1000];
    ULONG ulNumRead;
    HRESULT hr;
    if( pIXMLOutput == NULL )
        return;
    // Read from the stream.
    ZeroMemory( szBuf, 1000 );
    while( ( hr = pIXMLOutput->Read( szBuf, 1000, &ulNumRead ) ) =
    {
        cout << szBuf;
     }
} // Process resultset.</pre>
```

## **Passing Parameters to Templates**

This example shows how parameter values can be passed to XML commands. This XML template is specified as a command:

```
<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>
<sql:header><sql:param name='CategoryName'>Confections</sql:para
<sql:query>select * from Categories where CategoryName = @Catego
</ROOT>
```

The template includes an SQL query. The query requires a value for its parameter (**@CategoryName**). If no parameter value is passed, the default value (**Condiments**) is used.

In passing parameter values to a template, the parameter name and value both

must be specified.

This is the code:

private:

#define UNICODE #define \_UNICODE #define DBINITCONSTANTS #define INITGUID

#include <stdio.h>
#include <tchar.h>
#include <stddef.h>
#include <windows.h>
#include <iostream.h>
#include "oledb.h"
#include "SQLOLEDB.h"

HRESULT InitializeAndEstablishConnection(IDBInitialize \*\* ppIDBI HRESULT SetCommandProperties(ICommand \* pICommand); HRESULT ProcessResultSet(ISequentialStream \* pStreamOutput);

```
//-----
// Structure Definition Section
//------
struct COLUMNDATA
{
    DBLENGTH dwLength; // The length of the data field
    DBSTATUS dwStatus; // The status value
    BYTE bData[1]; // The start of the data field
};
class CSequentialStream : public ISequentialStream
{
```

ULONG	m_cRef;	// reference count
HANDLE	m_hFile;	// handle to the file
LPWSTR	m_wszFileN	Name; // the file name

public:

```
CSequentialStream( LPWSTR );
virtual ~CSequentialStream();
```

```
// IUnknown Methods
STDMETHODIMP_(ULONG) AddRef();
STDMETHODIMP_(ULONG) Release();
STDMETHODIMP QueryInterface( REFIID, LPVOID* );
```

```
// ISequentialStream Methods
STDMETHODIMP Read(
    /* [out] */ void __RPC_FAR*,
    /* [in] */ ULONG,
    /* [out] */ ULONG __RPC_FAR* );
STDMETHODIMP Write(
    /* [in] */ const void __RPC_FAR*,
    /* [in] */ ULONG,
    /* [out]*/ ULONG,
```

```
/* [out]*/ ULONG ___RPC_FAR* );
```

};

```
CSequentialStream::CSequentialStream
(
LPWSTR wszFileName
)
:
m_cRef( 0 ),
m_hFile( NULL ),
m_wszFileName( NULL )
```

{

)

)

```
// The constructor AddRef's.
  AddRef();
  // Allocate memory for the file name.
  m_wszFileName = (LPWSTR) CoTaskMemAlloc( ( wcslen( wszFil
  // Copy the file name
  wcscpy( m_wszFileName, wszFileName );
}
CSequentialStream::~CSequentialStream
(
{
  // Free any allocated memory.
  if( m_wszFileName )
    CoTaskMemFree( m_wszFileName );
  // Close the file.
  if( m_hFile )
    CloseHandle( m_hFile );
}
ULONG
CSequentialStream::AddRef
(
{
  return ++m_cRef;
}
ULONG CSequentialStream::Release()
{
```

```
if(--m_cRef)
    return m_cRef;
  delete this;
  return 0;
}
HRESULT CSequentialStream::QueryInterface
(
  REFIID riid,
  void** ppv
)
{
  *ppv = NULL;
  if (riid == IID_IUnknown)
    *ppv = this;
  if (riid == IID_ISequentialStream)
    *ppv = this;
  if(*ppv)
  {
    ((IUnknown*)*ppv)->AddRef();
    return S_OK;
  }
  return E_NOINTERFACE;
}
HRESULT CSequentialStream::Read
(
  void *pv,
```

```
ULONG cb,
ULONG* pcbRead
ULONG cBytesRead = 0;
// Parameter checking.
if(pcbRead)
  *pcbRead = 0;
if(!pv)
  return STG E INVALIDPOINTER;
if(cb == 0)
  return S_OK;
// Do we need to open the file?
if( m_hFile == NULL )
{
  // Open the file.
  m_hFile = CreateFile( m_wszFileName, GENERIC_READ, 0, N
  // If we failed to open the file, return E_FAIL.
  if( m_hFile == INVALID_HANDLE_VALUE )
    return E_FAIL;
}
// Clear the buffer.
ZeroMemory( pv, cb );
// Read cb bytes from the stream.
if( !ReadFile( m_hFile, pv, cb, &cBytesRead, NULL ) )
  return E_FAIL;
```

) {

```
// Inform the user how many bytes were read.
  if( NULL != pcbRead )
    *pcbRead = cBytesRead;
  if(cb != cBytesRead)
    return S FALSE;
  return S_OK;
}
HRESULT CSequentialStream::Write
(
  const void *pv,
  ULONG cb,
  ULONG* pcbWritten
)
{
  // For purposes of this example, only a read-only stream is needed.
  return STG E CANTSAVE;
}
void main()
{
  HRESULT
                    hr = S OK;
                  * pIDBInitialize = NULL;
  IDBInitialize
                     * pIDBCreateSession
                                           = NULL;
  IDBCreateSession
  IDBCreateCommand
                       * pIDBCreateCommand
                                                = NULL;
                   * pICommand
  ICommand
                                        = NULL:
  ICommandStream
                      * pICommandStream
                                             = NULL;
  ICommandWithParameters* pICommandWithParameters = NULL;
                     * pICommandText
  ICommandText
                                           = NULL;
                 * pIAccessor
  IAccessor
                                    = NULL;
                     nParams = 1;
  const ULONG
```

```
* pParams
                                  = NULL;
DBPARAMS
DBPARAMS
                  params;
                  acDBBinding[nParams];
DBBINDING
                    acDBBindStatus[nParams];
DBBINDSTATUS
DBORDINAL
                   rgParamOrdinals[1];
DBPARAMBINDINFO
                       rgParamBindInfo[1];
const WCHAR
                 * wszParamName =
                                    L"@CategoryName";
                 * wszDataSourceType = L"DBTYPE_WCHA
const WCHAR
BYTE
               sprocparams[1000];
                   * pCol = (COLUMNDATA *) sprocparams
COLUMNDATA
ISequentialStream
                 * pStreamOutput
                                  = NULL;
                     * pDBColumnInfo
DBCOLUMNINFO
                                       = NULL;
HACCESSOR
                   hAccessor;
CSequentialStream
                  XMLInput( L"TemplateFile.xml" );
```

```
typedef struct tagSPROCPARAMS
{
```

```
WCHAR CategoryName[25];
```

```
} SPROCPARAMS;
```

CoInitialize(0);

```
// Call a function to initialize and establish connection.
```

if (FAILED(hr = InitializeAndEstablishConnection(&pIDBInitialize
 goto Cleanup;

```
{
```

```
cout << "Failed to obtain IDBCreateSession interface.\n";
goto Cleanup;</pre>
```

}

```
if(FAILED(hr = pIDBCreateSession->CreateSession(
                  NULL,
                  IID IDBCreateCommand,
                  (IUnknown**) &pIDBCreateCommand)))
{
  cout << "pIDBCreateSession->CreateSession failed.\n";
  goto Cleanup;
}
//Access the ICommand interface.
if(FAILED(hr = pIDBCreateCommand->CreateCommand(
                  NULL,
                  IID_ICommand,
                  (IUnknown**) &pICommand)))
{
  cout << "Failed to access ICommand interface.\n";
  goto Cleanup;
}
// Get an ICommandStream interface
if(FAILED(pICommand->QueryInterface( IID_ICommandStream, (
{
  cout << "Failed to get an ICommandStream interface.\n";
  goto Cleanup;
}
//Use SetCommandStream() to specify the command stream.
if(FAILED(hr = pICommandStream->SetCommandStream(IID_ISe
{
  cout << "Failed to set command stream.\n";</pre>
  goto Cleanup;
```

}

```
// Set the command properties.
```

```
if (FAILED(hr = SetCommandProperties(pICommand)))
  goto Cleanup;
```

```
pCol->dwStatus = DBSTATUS_S_OK;
wcscpy( (LPWSTR) pCol->bData, L"Condiments" );
pCol->dwLength = wcslen( (LPWSTR) pCol->bData) * sizeof(WCl
```

/\*Describe the consumer buffer by filling in the array.

```
of DBBINDING structures. Each binding associates
```

a single parameter to the consumer's buffer.\*/

acDBBinding[0].iOrdinal = 1; acDBBinding[0].obLength = offsetof( COLUMNDATA, dwLeng = offsetof( COLUMNDATA, dwStatus acDBBinding[0].obStatus acDBBinding[0].pTypeInfo = NULL: acDBBinding[0].pObject = NULL: acDBBinding[0].pBindExt = NULL: acDBBinding[0].dwPart = DBPART\_VALUE | DBPART\_STAT acDBBinding[0].dwMemOwner = DBMEMOWNER\_CLIENTOV acDBBinding[0].dwFlags = 0: acDBBinding[0].bScale = 0; acDBBinding[0].obValue = offsetof( COLUMNDATA, bData ); acDBBinding[0].eParamIO = DBPARAMIO INPUT; acDBBinding[0].cbMaxLen = 50; = DBTYPE\_WSTR; acDBBinding[0].wType acDBBinding[0].bPrecision = 0;

rgParamOrdinals[0] = 1; rgParamBindInfo[0].bPrecision = 0; rgParamBindInfo[0].bScale = 0;

```
rgParamBindInfo[0].dwFlags = DBPARAMFLAGS_ISINPUT;
rgParamBindInfo[0].pwszDataSourceType = (WCHAR *)wszDataS
rgParamBindInfo[0].pwszName = (WCHAR *)wszParamName;
rgParamBindInfo[0].ulParamSize = 35;
```

```
if (FAILED(hr = pICommandStream->QueryInterface(
           IID ICommandWithParameters,
           (LPVOID *)&pICommandWithParameters)))
{
  cout << "Error.\n";</pre>
  goto Cleanup;
}
if (FAILED(hr = pICommandWithParameters->SetParameterInfo(
           nParams,
           rgParamOrdinals,
           rgParamBindInfo)))
{
  cout << "Error.\n";</pre>
  goto Cleanup;
}
//Create an accessor from the above set of bindings.
if (FAILED(hr = pICommandStream->QueryInterface(
                  IID_IAccessor,
                  (void**)&pIAccessor)))
{
  cout << "Failed to get IAccessor interface.\n";
  goto Cleanup;
}
if (FAILED(hr = pIAccessor->CreateAccessor(
              DBACCESSOR_PARAMETERDATA,
```

```
nParams,
acDBBinding,
sizeof(SPROCPARAMS),
&hAccessor,
acDBBindStatus)))
```

```
{
```

cout << "Failed to create accessor for the defined parameters.\n"; goto Cleanup;

} /\*

Fill in DBPARAMS structure for the command execution. This structure specifies the parameter values in the command and is then passed to Execute.

\*/

```
params.pData = sprocparams; //pCol->bData; //sprocparams;
params.cParamSets = 1;
params.hAccessor = hAccessor;
```

```
pParams = &params;
```

```
}
```

//Process the result set.

```
if (FAILED(hr = ProcessResultSet(pStreamOutput)))
{
   goto Cleanup;
}
```

Cleanup:

```
//Free up memory.
if( pStreamOutput )
    pStreamOutput->Release();
if (pICommandStream)
    pICommandStream->Release();
if (pIDBCreateCommand)
    pIDBCreateCommand->Release();
if (pIDBCreateSession)
    pIDBCreateSession->Release();
if (pIDBInitialize)
    pIDBInitialize>Release();
if (hr)
{
```

```
IErrorInfo* pErrorInfo = NULL;
BSTR bstrDesc = NULL;
GetErrorInfo(0, &pErrorInfo);
if (pErrorInfo)
{
    pErrorInfo->GetDescription(&bstrDesc);
    printf ("\r\nError: %S\r\n", bstrDesc ? bstrDesc : L"Unknown e
    SysFreeString(bstrDesc);
    pErrorInfo->Release();
  }
}
```

```
//Release the COM library.
  CoUninitialize();
};
//_____
HRESULT InitializeAndEstablishConnection(IDBInitialize ** ppIDBI
{
  HRESULT hr = S OK;
  IDBInitialize * pIDBInitialize = NULL;
  IDBProperties * pIDBProperties = NULL;
  DBPROP
               rgIDBProps[4];
  DBPROPSET
                 rgIDBPropSet[1];
           ii;
  int
  if (!ppIDBInitialize)
    return E_INVALIDARG;
  *ppIDBInitialize = NULL;
  /*
  Initialize the property values needed
  to establish the connection.
  */
  for(ii = 0; ii < 4; ii++)
```

```
VariantInit(&rgIDBProps[ii].vValue);
```

```
{
    printf("Failed to get IDBInitialize interface.\n");
    goto Cleanup;
}
//Server name.
rgIDBProps[0].dwPropertyID = DBPROP_INIT_DATASOURCE;
rgIDBProps[0].vValue.vt = VT_BSTR;
rgIDBProps[0].vValue.bstrVal= SysAllocString(L"server");
rgIDBProps[0].dwOptions = DBPROPOPTIONS_REQUIRED;
rgIDBProps[0].colid = DB_NULLID;
```

```
//Database.
```

```
rgIDBProps[1].dwPropertyID = DBPROP_INIT_CATALOG;
rgIDBProps[1].vValue.vt = VT_BSTR;
rgIDBProps[1].vValue.bstrVal= SysAllocString(L"Northwind");
rgIDBProps[1].dwOptions = DBPROPOPTIONS_REQUIRED;
rgIDBProps[1].colid = DB_NULLID;
```

```
//User name (login).
```

```
rgIDBProps[2].dwPropertyID = DBPROP_AUTH_USERID;
rgIDBProps[2].vValue.vt = VT_BSTR;
rgIDBProps[2].vValue.bstrVal= SysAllocString(L"sa");
rgIDBProps[2].dwOptions = DBPROPOPTIONS_REQUIRED;
rgIDBProps[2].colid = DB_NULLID;
```

//Password.

- // rgIDBProps[3].dwPropertyID = DBPROP\_AUTH\_PASSWORD;
- // rgIDBProps[3].vValue.vt = VT\_BSTR;
- // rgIDBProps[3].vValue.bstrVal= SysAllocString(L"password");
- // rgIDBProps[3].dwOptions = DBPROPOPTIONS\_REQUIRED;
- // rgIDBProps[3].colid = DB\_NULLID;

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (rgIDBProps) to the SetProperties method. \*/

```
rgIDBPropSet[0].guidPropertySet = DBPROPSET_DBINIT;
```

```
// rgInitPropSet[0].cProperties = 4;
rgIDBPropSet[0].cProperties = 3;
rgIDBPropSet[0].rgProperties = rgIDBProps;
```

```
//Set initialization properties.
```

```
cout << "Failed to get IDBProperties interface.\n";
goto Cleanup;</pre>
```

```
}
```

{

```
if (FAILED(hr = pIDBProperties->SetProperties(1, rgIDBPropSet)))
{
```

```
cout << "Failed to set initialization properties.\n";</pre>
```

goto Cleanup;

```
} //end if
```

```
//Now establish the connection to the data source.
if(FAILED(hr = pIDBInitialize->Initialize()))
{
    cout << "Problem in establishing connection to the data source.\n'
    goto Cleanup;
}</pre>
```

```
*ppIDBInitialize = pIDBInitialize;
```

Cleanup:

```
for(ii = 0; ii < 4; ii++)
VariantClear(&rgIDBProps[ii].vValue);</pre>
```

```
if (pIDBProperties)
```

```
pIDBProperties->Release();
```

return hr;

```
} //End of InitializeAndEstablishConnection.
```

```
HRESULT SetCommandProperties(ICommand * pICommand)
{
    HRESULT hr = S_OK;
    DBPROP rgProps[1];
    DBPROP rgProps[1];
```

```
DBPROPSET rgPropSet[1];
ICommandProperties* pICommandProperties = NULL;
```

```
VariantInit(&rgProps[0].vValue);
```

```
//Server name.
```

```
rgProps[0].dwPropertyID = SSPROP_STREAM_BASEPATH;
rgProps[0].vValue.vt = VT_BSTR;
rgProps[0].vValue.bstrVal= SysAllocString(L"D:\\Test");
rgProps[0].dwOptions = DBPROPOPTIONS_REQUIRED;
rgProps[0].colid = DB_NULLID;
```

/\*

Now that the properties are set, construct the DBPROPSET structure (rgInitPropSet). The DBPROPSET structure is used to pass an array of DBPROP structures (rgProps) to the SetProperties method. \*/

```
rgPropSet[0].guidPropertySet = DBPROPSET_SQLSERVERSTRE
rgPropSet[0].cProperties = 1;
```

```
rgPropSet[0].rgProperties = rgProps;
```

```
// Get an ICommandProperties interface.
  if(FAILED(pICommand->QueryInterface( IID_ICommandPropertie
  {
    cout << "Failed to get an ICommandProperties interface.\n";</pre>
    goto Cleanup;
  }
  //Set initialization properties.
  if(FAILED(hr = pICommandProperties->SetProperties(1, rgPropSet
  {
    cout << "Failed to set command properties.\n";
    goto Cleanup;
  }
Cleanup:
  VariantClear(&rgProps[0].vValue);
  if (pICommandProperties)
    pICommandProperties->Release();
  return hr;
}
//_____
//Retrieve and display data resulting from a query.
HRESULT ProcessResultSet(ISequentialStream * pStreamOutput)
{
  CHAR szBuf[1000];
  ULONG ulNumRead;
  HRESULT hr;
  if( pStreamOutput == NULL )
```

```
return E_INVALIDARG;
```

```
// Read from the stream
ZeroMemory( szBuf, 1000 );
while( ( hr = pStreamOutput->Read( szBuf, 1000, &ulNumRead ) )
    {
        cout << szBuf;
        cout.flush();
    }
    return hr;
} //ProcessResultSet.</pre>
```

# **ODBC**

To use the Microsoft® SQL Server<sup>™</sup> 2000 ODBC driver, you must be able to create ODBC data sources and ensure that the server has the correct version of the catalog stored procedures. To code an ODBC application that uses SQL Server, you must know how to allocate ODBC handles, set attributes, connect to an instance of SQL Server, execute queries, and process results.

# **Configuring the SQL Server ODBC Driver (ODBC)**

Before using ODBC applications with Microsoft® SQL Server<sup>TM</sup> 2000, you must know how to upgrade the version of the catalog stored procedures on earlier versions of SQL Server and add, delete, and test data sources.

# How to add a data source (ODBC)

You can add a data source by using ODBC Administrator, programmatically (by using **SQLConfigDataSource**), or by creating a file.

To add a data source by using ODBC Administrator

- 1. On the **Start** menu, point to **Settings**, and then click **Control Panel**.
- 2. Double-click **ODBC**.
- 3. Click the **User DSN**, **System DSN**, or **File DSN** tab, and then click **Add**.
- 4. Click **SQL Server**, and then click **Finish**.
- 5. Complete the Steps in the Create a New Data Source to SQL Server Wizard.

### To add a data source programmatically

• Call **SQLConfigDataSource** with the *fOption* set to either ODBC\_ADD\_DSN or ODBC\_ADD\_SYS\_DSN.

### To add a file data source

• Call **SQLDriverConnect** with a SAVEFILE=*file\_name* parameter in the connect string. If the connect is successful, the ODBC driver creates a file data source with the connection parameters in the location pointed to by the SAVEFILE parameter.

### Examples

### A. Create a data source using SQLConfigDataSource

#include <stdio.h>

#include <windows.h>
#include "sql.h"
#include <odbcinst.h>

int main()
{
RETCODE retcode;

```
UCHAR *szDriver = "SQL Server";
UCHAR *szAttributes =
"DSN=MyDSN\0DESCRIPTION=SQLConfigDSN Sample\0"
"SERVER=MySQL\0ADDRESS=MyServer\0NETWORK=dbmssocn"
"DATABASE=pubs\0";
```

retcode = SQLConfigDataSource(NULL, ODBC\_ADD\_DSN, szDriver, szAttributes);

### **B.** Create a file data source

Use the SAVEFILE keyword in **SQLDriverConnect** to create a file data source, and then use **SQLDriverConnect** to connect with the file data source. This example has been simplified by removing error handling.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

#define MAXBUFLEN 255

SQLHENVhenv = SQL\_NULL\_HENV;SQLHDBChdbc1 = SQL\_NULL\_HDBC;

int main() {

RETCODE retcode;

// This format of the SAVEFILE keyword saves a successful
// connection as the file Myfiledsn.dsn in the ODBC default
// directory for file DSNs.
SQLCHAR szConnStrIn[MAXBUFLEN] =
 "SAVEFILE=MyFileDSN;DRIVER={SQL Server};SERVER=
 "NETWORK=dbmssocn;UID=sa;PWD=MyPassWord;";

SQLCHAR szConnStrOut[MAXBUFLEN]; SQLSMALLINT cbConnStrOut = 0;

// Allocate the ODBC Environment and save handle.
retcode = SQLAllocHandle (SQL\_HANDLE\_ENV, NULL, &henv);

// Allocate an ODBC connection handle and connect. retcode = SQLAllocHandle(SQL\_HANDLE\_DBC, henv, &hdbc1); retcode = SQLDriverConnect(hdbc1, // Connection handle NULL, // Window handle szConnStrIn, // Input connect string SQL\_NTS, // Null-terminated string szConnStrOut, // Addr of output buffer MAXBUFLEN, // Size of output buffer &cbConnStrOut, // Address of output length SQL\_DRIVER\_NOPROMPT);

szConnStrOut, // Addr of output buffer MAXBUFLEN, // Size of output buffer &cbConnStrOut, // Address of output length SQL\_DRIVER\_NOPROMPT);

```
/* Clean up. */
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
}
```

See Also

Adding or Deleting an ODBC Data Source

# How to delete a data source (ODBC)

You can delete a data source by using ODBC Administrator, programmatically (by using **SQLConfigDataSource**), or by deleting a file.

To delete a data source by using ODBC Administrator

- 1. On the **Start** menu, point to **Settings**, and then click **Control Panel**.
- 2. Double-click 32bit **ODBC**.
- 3. Click the **User DSN**, **System DSN**, or **File DSN** tab.
- 4. Click the data source to delete.
- 5. Click **Remove**, and then confirm the deletion.

### To delete a user or system data source programmatically

• Call **SQLConfigDataSource** with the **fOption** parameter set to either ODBC\_REMOVE\_DSN or ODBC\_REMOVE\_SYS\_DSN.

### To delete a file data source

- 1. On the **Start** menu, point to **Settings**, and then click **Control Panel**.
- 2. Double-click 32bit **ODBC**.
- 3. Click the **File DSN** tab.
- 4. Click the file DSN to delete.
- 5. Click **Remove**.

# Examples

This example shows data source removal by using **SQLConfigDataSource**. It has been simplified by removing error checking.

```
#include <stdio.h>
#include <windows.h>
#include "sql.h"
#include <odbcinst.h>
```

int main() {

RETCODE retcode;

UCHAR \*szDriver = "SQL Server"; UCHAR \*szAttributes = "DSN=MyFileDSN";

retcode = SQLConfigDataSource(NULL, ODBC\_REMOVE\_DSN, szDriver, szAttributes);

}

See Also

**Deleting a Data Source** 

# How to upgrade the catalog stored procedures

This procedure is needed only when:

- Running a new Microsoft® SQL Server<sup>™</sup> 2000 ODBC driver against an earlier version of SQL Server.
- Running a new SQL Server OLE DB provider against an earlier version of SQL Server.
- Referencing an earlier version of SQL Server in an **sp\_addlinkedserver**, OPENROWSET, or OPENQUERY statement running on a new version of SQL Server. These statements use the SQL Server OLE DB provider to access the target SQL Server.

To ensure the proper operation of the SQL Server OLE DB provider or SQL Server ODBC driver, you must use the Instcat.sql script that comes with your new version of SQL Server to upgrade the catalog stored procedures on the earlier version of SQL Server. For example, when running the SQL Server version 7.0 ODBC driver against SQL Server 6.5, you must run the SQL Server 7.0 version of Instcat.sql against SQL Server 6.5.

### To upgrade the catalog stored procedures

To upgrade the catalog stored procedures, the system administrator runs a script by using the **isql** utility. To run **isql**, the computer must be installed as a client workstation for SQL Server. The system administrator should back up the **master** database before running Instcat.sql.

At a command prompt, use the **isql** utility to run the Instcat.sql script. For example:

C:> ISQL -Usa -Psa\_password -Sserver\_name -ilocation\Instcat.sql

# Arguments

sa\_password

Is the password of the system administrator.

### server\_name

Is the name of the server on which SQL Server resides.

### location

Is the full path of the location of Instcat.sql. You can use Instcat.sql from an installed SQL Server (the default location is C:\Mssql7\Install) or from the SQL Server compact disc (the default location is *D*:\*platform* where *D*: is the CD-ROM drive letter and *platform* is the appropriate server platform directory, such as 386).

The Instcat.sql script generates many messages. Most of these indicate how rows were affected by Transact-SQL statements issued by the script. These messages can be ignored, although the output should be scanned for messages that indicate an execution error. When Instcat.sql is run against a version 6.0 SQL Server, the message generated about the object sp\_MS\_upd\_sysobj\_category not existing can be ignored. The last message should indicate that Instcat.sql completed successfully.

The Instcat.sql script fails when there is not enough space available in the **master** database to store the catalog stored procedures or to log the changes to existing procedures. If the Instcat.sql script fails, contact your system administrator.

# See Also

Upgrading the Catalog Stored Procedures

# **Connecting to SQL Server (ODBC)**

Initializing an ODBC application involves allocating environment and connection handles, setting attributes for the handles to tailor the behavior of the driver and server, and then connecting to Microsoft® SQL Server<sup>™</sup> 2000.

# How to allocate handles and connect to SQL Server (ODBC)

### To allocate handles and connect to SQL Server

- 1. Include the ODBC header files Sql.h, Sqlext.h, Sqltypes.h.
- 2. Include the Microsoft® SQL Server<sup>™</sup> 2000 driver-specific header file, Odbcss.h.
- 3. Call **SQLAllocHandle** with a *HandleType* of SQL\_HANDLE\_ENV to initialize ODBC and allocate an environment handle.
- 4. Call **SQLSetEnvAttr** with *Attribute* set to SQL\_ATTR\_ODBC\_VERSION and *ValuePtr* set to SQL\_OV\_ODBC3 to indicate the application will use ODBC 3.*x*-format function calls.
- 5. Optionally, call **SQLSetEnvAttr** to set other environment options or **SQLGetEnvAttr** to get environment options.
- 6. Call **SQLAllocHandle** with a *HandleType* of SQL\_HANDLE\_DBC to allocate a connection handle.
- 7. Optionally, call **SQLSetConnectAttr** to set connection options or **SQLGetConnectAttr** to get connection options.
- 8. Call **SQLConnect** to use an existing data source to connect to SQL Server.

Or

Call **SQLDriverConnect** to use a connection string to connect to SQL

Server.

A minimum complete SQL Server connection string has one of two forms:

### DSN=dsn\_name;UID=login\_id;PWD=password;

# DRIVER={SQL Server};SERVER=server;UID=login\_id;PWD=password;

If the connection string is not complete, **SQLDriverConnect** can prompt for the required information. This is controlled by the value specified for the *DriverCompletion* parameter.

Or

Call **SQLBrowseConnect** multiple times in an iterative fashion to build the connection string and connect to SQL Server.

- 9. Optionally, call **SQLGetInfo** to get driver attributes and behavior for the SQL Server data source.
- 10. Allocate and use statements.
- 11. Call **SQLDisconnect** to disconnect from SQL Server and make the connection handle available for a new connection.
- 12. Call **SQLFreeHandle** with a *HandleType* of SQL\_HANDLE\_DBC to free the connection handle.
- 13. Call **SQLFreeHandle** with a *HandleType* of SQL\_HANDLE\_ENV to free the environment handle.

# Examples

# A. Allocate handles, then connect by using SQLConnect

This example shows allocating an environment handle and a connection handle,

then connecting by using **SQLConnect**. It has been simplified by removing much of the error checking.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <sqlext.h>
#include <sqltypes.h>
#include <odbcss.h>
```

```
SQLHENVhenv = SQL_NULL_HENV;SQLHDBChdbc1 = SQL_NULL_HDBC;SQLHSTMThstmt1 = SQL_NULL_HSTMT;
```

int main() {

RETCODE retcode;

UCHAR szDSN[SQL\_MAX\_DSN\_LENGTH+1] = "MyDSN", szUID[MAXNAME] = "sa", szAuthStr[MAXNAME] = "MyPassword";

// Allocate the ODBC Environment and save handle.
retcode = SQLAllocHandle (SQL\_HANDLE\_ENV, NULL, &henv);

```
if ( (retcode != SQL_SUCCESS) &&
  (retcode != SQL_SUCCESS_WITH_INFO) ) {
    // Connect failed, call SQLGetDiagRec for errors.
}
else {
    // Connects to SQL Server always return
    // informational messages. These messages can be
    // retrieved by calling SQLGetDiagRec.
}
// Allocate statement handles and do ODBC processing.
/* Clean up. */
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
```

```
SQLFreeHandle(SQL_HANDLE_ENV, henv);
```

```
}
```

return(0);

### B. Connect to SQL Server without an existing ODBC data source

This example shows a call to **SQLDriverConnect** to connect to an instance of SQL Server without requiring an existing ODBC data source:

#define MAXBUFLEN 255

SQLHENV	henv = SQL_NULL_HENV;
SQLHDBC	hdbc1 = SQL_NULL_HDBC;
SQLHSTMT	hstmt1 = SQL_NULL_HSTMT;

```
SQLCHAR ConnStrIn[MAXBUFLEN] =
    "DRIVER={SQL Server};SERVER=MyServer;"
    "UID=sa;PWD=MyPassWord;DATABASE=pubs;";
```

### SQLCHAR ConnStrOut[MAXBUFLEN]; SQLSMALLINT cbConnStrOut = 0;

// Make connection without data source. Ask that driver not

// prompt if insufficient information. Driver returns

// SQL\_ERROR and application prompts user

// for missing information. Window handle not needed for

// SQL\_DRIVER\_NOPROMPT.

retcode = SQLDriverConnect(hdbc1, // Connection handle NULL, // Window handle ConnStrIn, // Input connect string SQL\_NTS, // Null-terminated string ConnStrOut, // Address of output buffer MAXBUFLEN, // Size of output buffer &cbConnStrOut, // Address of output length SQL\_DRIVER\_NOPROMPT);

### See Also

**SQLFreeHandle** 

<u>SQLGetInfo</u>

**SQLBrowseConnect** 

<u>SQLSetConnectAttr</u>

<u>SQLDriverConnect</u>

<u>SQLSetEnvAttr</u>

<u>SQLGetConnectAttr</u>

# **Executing Queries (ODBC)**

Executing an SQL statement in an ODBC application requires allocating a statement handle, setting statement attributes, and preparing and executing the SQL statement.

# How to use a statement (ODBC)

#### To use a statement

- 1. Call **SQLAllocHandle** with a *HandleType* of SQL\_HANDLE\_STMT to allocate a statement handle.
- 2. Optionally, call **SQLSetStmtAttr** to set statement options or **SQLGetStmtAttr** to get statement attributes.

To use server cursors, you must set cursor attributes to values other than their defaults.

- 3. Optionally, if the statement will be executed several times, prepare the statement for execution with **SQLPrepare**.
- 4. Optionally, if the statement has bound parameter markers, bind the parameter markers to program variables by using SQLBindParameter. If the statement was prepared, you can call SQLNumParams and SQLDescribeParam to find the number and characteristics of the parameters.
- 5. Execute a statement directly by using **SQLExecDirect**.

Or

If the statement was prepared, execute it multiple times by using **SQLExecute**.

Or

Call a catalog function, which returns results.

6. Process the results by binding the result set columns to program variables, by moving data from the result set columns to program variables by using **SQLGetData**, or a combination of the two methods.

Fetch through the result set of a statement one row at a time.

Or

Fetch through the result set several rows at a time by using a block cursor.

Or

Call **SQLRowCount** to determine the number of rows affected by an INSERT, UPDATE, or DELETE statement.

If the SQL statement can have multiple result sets, call **SQLMoreResults** at the end of each result set to see if there are additional result sets to process.

- 7. After results are processed, the following actions may be necessary to make the statement handle available to execute a new statement:
  - If you did not call **SQLMoreResults** until it returned SQL\_NO\_DATA, call **SQLCloseCursor** to close the cursor.
  - If you bound parameter markers to program variables, call **SQLFreeStmt** with *Option* set to SQL\_RESET\_PARAMS to free the bound parameters.
  - If you bound result set columns to program variables, call **SQLFreeStmt** with *Option* set to SQL\_UNBIND to free the bound columns.
  - To reuse the statement handle, go to Step 2.
- 8. Call **SQLFreeHandle** with a *HandleType* of SQL\_HANDLE\_STMT to free the statement handle.

### See Also

Allocating a Statement Handle

Constructing an SQL Statement

**Direct Execution** 

Freeing a Statement Handle

Prepared Execution

**SQLBindParameter** 

**SQLDescribeParam** 

**SQLFreeHandle** 

<u>SQLGetData</u>

<u>SQLGetStmtAttr</u>

**SQLMoreResults** 

**SQLRowCount** 

<u>SQLSetStmtAttr</u>

# How to set cursor options (ODBC)

#### To set cursor options

• Call **SQLSetStmtAttr** to set or **SQLGetStmtAttr** to get the statement options that control cursor behavior.

Foption	Specifies
SQL_ATTR_CURSOR_TYPE	Cursor type of forward-
	only, static, dynamic, or keyset-driven
SQL_ATTR_CONCURRENCY	Concurrency control
	option of read-only,
	locking, optimistic using
	timestamps, or optimistic
	using values
SQL_ATTR_ROW_ARRAY_SIZE	Number of rows retrieved
	in each fetch
SQL_ATTR_CURSOR_SENSITIVITY	Cursor that does or does
	not show updates to
	cursor rows made by
	other connections
SQL_ATTR_CURSOR_SCROLLABLE Cursor that can be	
	scrolled forward and
	backward

The default values for these attributes (forward-only, read-only, rowset size of 1) do not use server cursors. To use server cursors, at least one of these attributes must be set to a value other than the default, and the statement being executed must be a single SELECT statement or a stored procedure that contains a single SELECT statement. When using server cursors, SELECT statements cannot use clauses not supported by server cursors: COMPUTE, COMPUTE BY, FOR BROWSE, and INTO.

You can control the type of cursor used either by setting SQL\_ATTR\_CURSOR\_TYPE and SQL\_ATTR\_CONCURRENCY, or by setting SQL\_ATTR\_CURSOR\_SENSITIVITY and SQL\_ATTR\_CURSOR\_SCROLLABLE. You should not mix the two methods of specifying cursor behavior.

# Examples

# A. Allocate a statement handle, set a dynamic cursor type with row versioning optimistic concurrency, and then execute a SELECT

```
SQL_NTS);
```

# **B.** Allocate a statement handle, set a scrollable, sensitive cursor, and then execute a SELECT

```
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt1)
// Set the cursor options and execute the statement.
```

retcode = SQLSetStmtAttr(hstmt1, SQL\_ATTR\_CURSOR\_SCROLL/ (SQLPOINTER)SQL\_SCROLLABLE,

SQL\_IS\_INTEGER);

retcode = SQLSetStmtAttr(hstmt1, SQL\_ATTR\_CURSOR\_SENSITIV (SQLPOINTER)SQL\_INSENSITIVE,

```
SQL_IS_INTEGER);
```

```
retcode = SQLExecDirect(hstmt1,
```

"select au\_lname from authors",
SQL\_NTS);

# See Also

Constructing SQL Statements for Cursors

<u>SQLGetStmtAttr</u>

<u>SQLSetStmtAttr</u>

# How to execute a statement directly (ODBC)

### To execute a statement directly and one time only

- 1. If the statement has parameter markers, use **SQLBindParameter** to bind each parameter to a program variable. Fill the program variables with data values, and then set up any data-at-execution parameters.
- 2. Call **SQLExecDirect** to execute the statement.
- 3. If data-at-execution input parameters are used, **SQLExecDirect** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

# To execute a statement multiple times by using column-wise parameter binding

- 1. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_PARAMSET\_SIZE to the number of sets (S) of parameters.
  - Set SQL\_ATTR\_PARAM\_BIND\_TYPE to SQL\_PARAMETER\_BIND\_BY\_COLUMN.
  - Set the SQL\_ATTR\_PARAMS\_PROCESSED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of parameters processed.
  - Set SQL\_ATTR\_PARAMS\_STATUS\_PTR to point to an array[S] of SQLUSSMALLINT variables to hold the parameter status indicators.
- 2. For each parameter marker:

- Allocate an array of S parameter buffers to store data values.
- Allocate an array of S parameter buffers to store data lengths.
- Call **SQLBindParameter** to bind the parameter data value and data length arrays to the statement parameter.
- Set up any data-at-execution **text** or **image** parameters.
- Put S data values and S data lengths into the bound parameter arrays.
- 3. Call **SQLExecDirect** to execute the statement. The driver efficiently executes the statement S times, once for each set of parameters.
- 4. If data-at-execution input parameters are used, **SQLExecDirect** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

### To execute a statement multiple times by using row-wise parameter binding

- 1. Allocate an array[S] of structures, where S is the number of sets of parameters. The structure has one element for each parameter, and each element has two parts:
  - The first part is a variable of the appropriate data type to hold the parameter data.
  - The second part is a SQLINTEGER variable to hold the status indicator.
- 2. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_PARAMSET\_SIZE to the number of sets (S) of parameters.

- Set SQL\_ATTR\_PARAM\_BIND\_TYPE to the size of the structure allocated in Step 1.
- Set the SQL\_ATTR\_PARAMS\_PROCESSED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of parameters processed.
- Set SQL\_ATTR\_PARAMS\_STATUS\_PTR to point to an array[S] of SQLUSSMALLINT variables to hold the parameter status indicators.
- 3. For each parameter marker, call **SQLBindParameter** to point the parameter's data value and data length pointer to their variables in the first element of the array of structures allocated in Step 1. If the parameter is a data-at-execution parameter, set it up.
- 4. Fill the bound parameter buffer array with data values.
- 5. Call **SQLExecDirect** to execute the statement. The driver efficiently executes the statement S times, once for each set of parameters.
- 6. If data-at-execution input parameters are used, **SQLExecDirect** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

Column-wise and row-wise binding are more typically used in conjunction with **SQLPrepare** and **SQLExecute** than with **SQLExecDirect**.

## Examples

This example shows executing a SELECT statement by using **SQLExecDirect**. It has been simplified by removing all error checking.

```
#include <stdio.h>
#include <string.h>
```

```
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

#define MAXBUFLEN 255

```
SQLHENVhenv = SQL_NULL_HENV;SQLHDBChdbc1 = SQL_NULL_HDBC;SQLHSTMThstmt1 = SQL_NULL_HSTMT;
```

```
int main()
```

```
{
```

```
RETCODE retcode;
// SQLBindCol variables
SQLCHAR szName[MAXNAME+1];
SQLINTEGER cbName;
```

```
"MyPassWord", SQL_NTS);
```

// Allocate a statement handle.

retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt)

// Execute an SQL statement directly on the statement handle.

```
// Uses a default result set because no cursor attributes are set.
retcode = SQLExecDirect(hstmt1,
```

```
"SELECT au_lname FROM authors",
    SQL_NTS);
// Simplified result set processing. Bind one column and
// then fetch until SQL_NO_DATA.
retcode = SQLBindCol(hstmt1, 1, SQL_C_CHAR,
    szName, MAXNAME, &cbName);
while ( (retcode = SQLFetch(hstmt1) ) != SQL_NO_DATA )
    printf("Name = %s\n", szName);
```

```
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
```

## See Also

}

**Binding Parameters** 

**Direct Execution** 

**SQLBindParameter** 

SQLPutData

<u>SQLSetStmtAttr</u>

**Using Statement Parameters** 

## How to prepare and execute a statement (ODBC)

### To prepare a statement once, and then execute it multiple times

- 1. Call **SQLPrepare** to prepare the statement.
- 2. Optionally, call **SQLNumParams** to determine the number of parameters in the prepared statement.
- 3. Optionally, for each parameter in the prepared statement:
  - Call **SQLDescribeParam** to get parameter information.
  - Bind each parameter to a program variable by using **SQLBindParam**. Set up any data-at-execution parameters.
- 4. For each execution of a prepared statement:
  - If the statement has parameter markers, put the data values into the bound parameter buffer.
  - Call **SQLExecute** to execute the prepared statement.
  - If data-at-execution input parameters are used, **SQLExecute** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

#### To prepare a statement with column-wise parameter binding

- 1. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_PARAMSET\_SIZE to the number of sets (S) of parameters.
  - Set SQL\_ATTR\_PARAM\_BIND\_TYPE to

### SQL\_PARAMETER\_BIND\_BY\_COLUMN.

- Set the SQL\_ATTR\_PARAMS\_PROCESSED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of parameters processed.
- Set SQL\_ATTR\_PARAMS\_STATUS\_PTR to point to an array[S] of SQLUSSMALLINT variables to hold parameter status indicators.
- 2. Call **SQLPrepare** to prepare the statement.
- 3. Optionally, call **SQLNumParams** to determine the number of parameters in the prepared statement.
- 4. Optionally, for each parameter in the prepared statement, call **SQLDescribeParam** to get parameter information.
- 5. For each parameter marker:
  - Allocate an array of S parameter buffers to store data values.
  - Allocate an array of S parameter buffers to store data lengths.
  - Call **SQLBindParameter** to bind the parameter data value and data length arrays to the statement parameter.
  - If the parameter is a data-at-execution **text** or **image** parameter, set it up.
  - If any data-at-execution parameters are used, set them up.
- 6. For each execution of a prepared statement:

- Put the S data values and S data lengths into the bound parameter arrays.
- Call **SQLExecute** to execute the prepared statement.
- If data-at-execution input parameters are used, **SQLExecute** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

### To prepare a statement with row-wise bound parameters

- 1. Allocate an array[S] of structures, where S is the number of sets of parameters. The structure has one element for each parameter, and each element has two parts:
  - The first part is a variable of the appropriate data type to hold the parameter data.
  - The second part is a SQLINTEGER variable to hold the status indicator.
- 2. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_PARAMSET\_SIZE to the number of sets (S) of parameters.
  - Set SQL\_ATTR\_PARAM\_BIND\_TYPE to the size of the structure allocated in Step 1.
  - Set the SQL\_ATTR\_PARAMS\_PROCESSED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of parameters processed.
  - Set SQL\_ATTR\_PARAMS\_STATUS\_PTR to point to an array[S] of SQLUSSMALLINT variables to hold parameter status indicators.

- 3. Call **SQLPrepare** to prepare the statement.
- 4. For each parameter marker, call **SQLBindParameter** to point the parameter data value and data length pointer to their variables in the first element of the array of structures allocated in Step 1. If the parameter is a data-at-execution parameter, set it up.
- 5. For each execution of a prepared statement:
  - Fill the bound parameter buffer array with data values.
  - Call **SQLExecute** to execute the prepared statement. The driver efficiently executes the SQL statement S times, once for each set of parameters.
  - If data-at-execution input parameters are used, **SQLExecute** returns SQL\_NEED\_DATA. Send the data in chunks by using **SQLParamData** and **SQLPutData**.

## Examples

This example shows executing a SELECT statement by using **SQLPrepare** and **SQLExecute**. It has been simplified by removing all error checking.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

#define MAXBUFLEN 255

SQLHENV henv = SQL\_NULL\_HENV;

```
SQLHDBC hdbc1 = SQL_NULL_HDBC;
SQLHSTMT hstmt1 = SQL_NULL_HSTMT;
```

```
int main()
```

{

RETCODE retcode; // SQLBindCol variables SQLCHAR szName[MAXNAME+1]; SQLINTEGER cbName;

```
"MyPassWord", SQL_NTS);
```

// Allocate a statement handle.

retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt)
// Prepare and execute an SQL statement on the statement handle.
// Uses a default result set because no cursor attributes are set.
retcode = SQLPrepare(hstmt1,

"SELECT au\_lname from authors", SQL\_NTS);
retcode = SQLExecute(hstmt1);

```
// Simplified result set processing. Bind one column and
// then fetch until SQL_NO_DATA.
```

```
retcode = SQLBindCol(hstmt1, 1, SQL_C_CHAR,
```

szName, MAXNAME, &cbName);

while ( ( retcode = SQLFetch(hstmt1) ) != SQL\_NO\_DATA )

```
printf("Name = %s\n", szName);
```

```
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
```

```
See Also
```

}

```
Binding Parameters
SQLBindParameter
SQLDescribeParam
SQLPrepare
SQLPutData
SQLSetStmtAttr
Prepared Execution
Using Statement Parameters
```

# **Processing Results (ODBC)**

Processing results in an ODBC application involves first determining the characteristics of the result set, then retrieving the data into program variables by using either **SQLBindCol** or **SQLGetData**.

## How to retrieve result set information (ODBC)

#### To get information about a result set

- 1. Call **SQLNumResultCols** to get the number of columns in the result set.
- 2. For each column in the result set:
  - Call SQLDescribeCol to get information about the result column.

Or

• Call SQLColAttribute to get specific descriptor information about the result column.

### See Also

Determining the Characteristics of a Result Set

How to process results (ODBC)

**SQLColAttribute** 

**SQLDescribeCol** 

<u>SQLNumResultCols</u>

## How to process results (ODBC)

#### **To process results**

- 1. Retrieve result set information.
- 2. If bound columns are used, for each column you want to bind to, call **SQLBindCol** to bind a program buffer to the column.
- 3. For each row in the result set:
  - Call **SQLFetch** to get the next row.
  - If bound columns are used, use the data now available in the bound column buffers.
  - If unbound columns are used, call **SQLGetData** one or more times to get the data for unbound columns after the last bound column. Calls to **SQLGetData** should be in increasing order of column number.
  - Call **SQLGetData** multiple times to get data from a **text** or **image** column.
- 4. When **SQLFetch** signals the end of the result set by returning SQL\_NO\_DATA, call **SQLMoreResults** to determine if another result set is available.
  - If it returns SQL\_SUCCESS, another result set is available.
  - If it returns SQL\_NO\_DATA, no more result sets are available.

• If it returns SQL\_SUCCESS\_WITH\_INFO or SQL\_ERROR, call **SQLGetDiagRec** to determine if the output from a PRINT or RAISERROR statement is available.

If bound statement parameters are used for output parameters or the return value of a stored procedure, use the data now available in the bound parameter buffers. Also, when bound parameters are used, each call to **SQLExecute** or **SQLExecDirect** will have executed the SQL statement S times, where S is the number of elements in the array of bound parameters. This means that there will be S sets of results to process, where each set of results comprises all of the result sets, output parameters, and return codes usually returned by a single execution of the SQL statement.

Note that when a result set contains compute rows, each compute row is made available as a separate result set. These compute result sets are interspersed within the normal rows and break normal rows into multiple result sets.

- 5. Optionally, call **SQLFreeStmt** with an *fOption* of SQL\_UNBIND to release any bound column buffers.
- 6. If another result set is available, go to Step 1.

To cancel processing a result set before **SQLFetch** returns SQL\_NO\_DATA, call **SQLCloseCursor**.

## Examples

This example shows how to use either **SQLBindCol** or **SQLGetData**. It has been simplified by removing all error checking. The program can be compiled with either the **SQLBindCol** function or the **SQLGetData** function commented out, the resulting executable will operate the same.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
```

#include <sqlext.h>
#include <odbcss.h>

#define MAXBUFLEN 255

SQLHENVhenv = SQL\_NULL\_HENV;SQLHDBChdbc1 = SQL\_NULL\_HDBC;SQLHSTMThstmt1 = SQL\_NULL\_HSTMT;

int main() {
 RETCODE retcode;
 // SQLBindCol variables
 SQLCHAR szName[MAXNAME+1];
 SQLINTEGER cbName;

// Allocate a statement handle.

retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt: // Execute an SQL statement directly on the statement handle.

// Uses a default result set because no cursor attributes are set.
retcode = SQLExecDirect(hstmt1,

"SELECT au\_lname FROM authors", SQL\_NTS);

// Simplified result set processing. Fetch until SQL\_NO\_DATA.

// The application can be compiled with the SQLBindCol line // commented out to illustrate SQLGetData, or compiled with the // SQLGetData line commented out to illustrate SQLBindCol. // This sample shows that SQLBindCol is called once for the // result set, while SQLGetData must be called once for each // row in the result set.

```
/* Clean up.*/
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
```

return(0);
}

## See Also

Assigning Storage (Binding) Determining the Characteristics of a Result Set Fetching Result Data How to retrieve result set information (ODBC) SQLBindCol

<u>SQLCloseCursor</u>

<u>SQLFreeStmt</u> <u>SQLGetData</u>

**SQLMoreResults** 

# **Using Cursors (ODBC)**

To use cursors, you must first set connection and statement attributes that control ODBC cursor behavior. Cursors allow an application to retrieve multiple rows on each fetch and execute UPDATE, INSERT, or DELETE statements at the current location of the cursor.

## How to use cursors (ODBC)

#### To use cursors

1. Call **SQLSetStmtAttr** to set the desired cursor attributes:

Set the SQL\_ATTR\_CURSOR\_TYPE and SQL\_ATTR\_CONCURRENCY attributes (this is the preferred option).

Or

Set the SQL\_CURSOR\_SCROLLABLE and SQL\_CURSOR\_SENSITIVITY attributes.

- 2. Call **SQLSetStmtAttr** to set the rowset size by using the SQL\_ATTR\_ROW\_ARRAY\_SIZE attribute.
- 3. Optionally, call **SQLSetCursorName** to set a cursor name if positioned updates will be done by using the WHERE CURRENT OF clause.
- 4. Execute the SQL statement.
- 5. Optionally, call **SQLGetCursorName** to get the cursor name if positioned updates will be done by using the WHERE CURRENT OF clause and a cursor name was not supplied with **SQLSetCursorName** in Step 3.
- 6. Call **SQLNumResultCols** to get the number of columns (C) in the rowset.
- 7. Use column-wise binding.

Or

Use row-wise binding.

- 8. Fetch rowsets from the cursor as desired.
- 9. Call **SQLMoreResults** to determine if another result set is available.
  - If it returns SQL\_SUCCESS, another result set is available.
  - If it returns SQL\_NO\_DATA, no more result sets are available.
  - If it returns SQL\_SUCCESS\_WITH\_INFO or SQL\_ERROR, call **SQLGetDiagRec** to determine if the output from a PRINT or RAISERROR statement is available.

If bound statement parameters are used for output parameters or the return value of a stored procedure, use the data now available in the bound parameter buffers.

When bound parameters are used, each call to **SQLExecute** or **SQLExecDirect** will have executed the SQL statement S times, where S is the number of elements in the array of bound parameters. This means that there will be S sets of results to process, where each set of results comprises all of the result sets, output parameters, and return codes usually returned by a single execution of the SQL statement.

Note that when a result set contains compute rows, each compute row is made available as a separate result set. These compute result sets are interspersed within the normal rows and break normal rows into multiple result sets.

- 10. Optionally, call **SQLFreeStmt** with an *fOption* of SQL\_UNBIND to release any bound column buffers.
- 11. If another result set is available, go to Step 6.

In Step 9, calling **SQLMoreResults** on a partially processed result set clears the remainder of the result set. Another way to clear a partially processed result set

#### is to call **SQLCloseCursor**.

You can control the type of cursor used by either setting SQL\_ATTR\_CURSOR\_TYPE and SQL\_ATTR\_CONCURRENCY, or by setting SQL\_ATTR\_CURSOR\_SENSITIVITY and SQL\_ATTR\_CURSOR\_SCROLLABLE. You should not mix the two methods of specifying cursor behavior.

### See Also

How Cursors Are Implemented

How to use rowset binding (ODBC)

**SQLFreeStmt** 

**SQLGetCursorName** 

**SQLMoreResults** 

**SQLNumResultCols** 

<u>SQLSetStmtAttr</u>

**Using Default Result Sets** 

## How to use rowset binding (ODBC)

#### To use column-wise binding

- 1. For each bound column
  - Allocate an array of R (or more) column buffers to store data values, where R is number of rows in the rowset.
  - Optionally, allocate an array of R (or more) column buffers to store data lengths.
  - Call **SQLBindCol** to bind the column's data value and data length arrays to the column of the rowset.
- 2. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_ROW\_ARRAY\_SIZE to the number of rows in the rowset (R).
  - Set SQL\_ATTR\_ROW\_BIND\_TYPE to SQL\_BIND\_BY\_COLUMN.
  - Set the SQL\_ATTR\_ROWS FETCHED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of rows fetched.
  - Set SQL\_ATTR\_ROW\_STATUS\_PTR to point to an array[R] of SQLUSSMALLINT variables to hold the row-status indicators.
- 3. Execute the statement.
- 4. Each call to **SQLFetch** or **SQLFetchScroll** retrieves R rows and

transfers the data into the bound columns.

#### To use row-wise binding

- 1. Allocate an array[R] of structures, where R is the number of rows in the rowset. The structure has one element for each column, and each element has two parts:
  - The first part is a variable of the appropriate data type to hold the column data.
  - The second part is a SQLINTEGER variable to hold the column status indicator.
- 2. Call **SQLSetStmtAttr** to set the following attributes:
  - Set SQL\_ATTR\_ROW\_ARRAY\_SIZE to the number of rows in the rowset (R).
  - Set SQL\_ATTR\_ROW\_BIND\_TYPE to the size of the structure allocated in Step 1.
  - Set the SQL\_ATTR\_ROWS\_FETCHED\_PTR attribute to point to a SQLUINTEGER variable to hold the number of rows fetched.
  - Set SQL\_ATTR\_PARAMS\_STATUS\_PTR to point to an array[R] of SQLUSSMALLINT variables to hold the row-status indicators.
- 3. For each column in the result set, call **SQLBindCol** to point the data value and data length pointer of the column to their variables in the first element of the array of structures allocated in Step 1.
- 4. Execute the statement.

5. Each call to **SQLFetch** or **SQLFetchScroll** retrieves R rows and transfers the data into the bound columns.

## See Also

How Cursors Are Implemented How to use cursors (ODBC) SQLBindCol SQLFetchScroll SQLSetStmtAttr Using Default Result Sets

## How to fetch and update rowsets (ODBC)

#### To fetch and update rowsets

- 1. Optionally, call **SQLSetStmtAttr** with an *fOption* of SQL\_ROW\_ARRAY\_SIZE to change the number of rows (R) in the rowset.
- 2. Call **SQLFetch** or **SQLFetchScroll** to get a rowset.
- 3. If bound columns are used, use the data values and data lengths now available in the bound column buffers for the rowset.

If unbound columns are used, for each row call **SQLSetPos** with *Operation* set to SQL\_POSITION to set the cursor position; then, for each unbound column:

- Call **SQLGetData** one or more times to get the data for unbound columns after the last bound column of the rowset. Calls to **SQLGetData** should be in order of increasing column number.
- Call **SQLGetData** multiple times to get data from a **text** or **image** column.
- 4. Set up any data-at-execution **text** or **image** columns.
- 5. Call **SQLSetPos** or **SQLBulkOperations** to set the cursor position, refresh, update, delete, or add row(s) within the rowset.

If data-at-execution **text** or **image** columns are used for an update or add operation, handle them.

6. Optionally, execute a positioned UPDATE or DELETE statement, specifying the cursor name (available from **SQLGetCursorName**) and using a different statement handle on the same connection.

## See Also

Bookmarking Rows Changing Rows with Positioned Operations Scrolling and Retrieving Rows SQLFetchScroll SQLGetCursorName SQLGetData SQLSetStmtAttr

# **Performing Transactions (ODBC)**

In ODBC, transactions cannot span connections. ODBC applications can use the standard ODBC transaction management functions to work with transactions on individual connections. ODBC applications can also use the Microsoft Distributed Transaction Coordinator (MS DTC) to include multiple Microsoft® SQL Server<sup>™</sup> connections in a single transaction, even when the connections are to separate servers.

# How to use Microsoft Distributed Transaction Coordinator (ODBC)

#### To update two or more SQL Servers by using MS DTC

- Connect to MS DTC by using the MS DTC OLE **DtcGetTransactionManager** function. For information about MS DTC, see Microsoft Distributed Transaction Coordinator.
- 2. Call **SQLDriverConnect** once for each Microsoft® SQL Server<sup>™</sup> connection you want to establish.
- 3. Call the MS DTC OLE **ITransactionDispenser::BeginTransaction** function to begin an MS DTC transaction and obtain a Transaction object that represents the transaction.
- Call SQLSetConnectAttr one or more times for each ODBC connection you want to enlist in the MS DTC transaction.
   SQLSetConnectAttr must be called with an *fOption* of SQL\_ATTR\_ENLIST\_IN\_DTC and a *vParam* of the Transaction object (obtained in Step 3).
- 5. Call **SQLExecDirect** once for each SQL Server you want to update.
- 6. Call the MS DTC OLE **ITransaction::Commit** function to commit the MS DTC transaction. The Transaction object is no longer valid.

To perform a series of MS DTC transactions, repeat Steps 3 through 6.

To release the reference to the Transaction object, call the MS DTC OLE **ITransaction::Return** function.

To use an ODBC connection with an MS DTC transaction, and then use the same connection with a local SQL Server transaction, call **SQLSetConnectAttr** 

with a *vParam* of SQL\_DTC\_DONE.

**Note** You can also call **SQLSetConnectAttr** and **SQLExecDirect** in turn for each SQL Server instead of calling them as suggested earlier in Steps 4 and 5.

## See Also

Performing Distributed Transactions

**SQLDriverConnect** 

<u>SQLSetConnectAttr</u>

# **Running Stored Procedures (ODBC)**

The Microsoft® SQL Server<sup>™</sup> ODBC driver supports executing stored procedures as remote stored procedures. Executing a stored procedure as a remote stored procedure allows the driver and the server to optimize the performance of executing the procedure.

# How to call stored procedures (ODBC)

When a SQL statement calls a stored procedure using the ODBC CALL escape clause, the Microsoft® SQL Server<sup>TM</sup> driver sends the procedure to SQL Server using the remote stored procedure call (RPC) mechanism. RPC requests bypass much of the statement parsing and parameter processing in SQL Server and are faster than using the Transact-SQL EXECUTE statement.

#### To run a procedure as an RPC

- 1. Construct a SQL statement that uses the ODBC CALL escape sequence. The statement uses parameter markers for each input, input/output, and output parameter, and for the procedure return value (if any): {? = CALL procname (?,?)}
- 2. Call **SQLBindParameter** for each input, input/output, and output parameter, and for the procedure return value (if any).
- 3. Execute the statement with **SQLExecDirect**.

**Note** If an application submits a procedure using the Transact-SQL EXECUTE syntax (as opposed to the ODBC CALL escape sequence), the SQL Server ODBC driver passes the procedure call to SQL Server as a SQL statement rather than as an RPC. Also, output parameters are not returned if the Transact-SQL EXECUTE statement is used.

## See Also

**Batching Stored Procedure Calls** 

Running Stored Procedures

Calling a Stored Procedure

<u>SQLBindParameter</u>

**Procedures** 

# How to process return codes and output parameters (ODBC)

Microsoft® SQL Server<sup>™</sup> stored procedures can have integer return codes and output parameters. The return codes and output parameters are sent in the last packet from the server and are not available to the application until **SQLMoreResults** returns SQL\_NO\_DATA.

#### To process return codes and output parameters

- 1. Construct a SQL statement that uses the ODBC CALL escape sequence. The statement should use parameter markers for each input, input/output, and output parameter, and for the procedure return value (if any).
- 2. Call **SQLBindParameter** for each input, input/output, and output parameter, and for the procedure return value (if any).
- 3. Execute the statement with **SQLExecDirect**.
- 4. Process result sets until SQLFetch or SQLFetchScroll returns SQL\_NO\_DATA while processing the last result set or until SQLMoreResults returns SQL\_NO\_DATA. At this point, the variables bound to the return code and output parameters are filled with returned data values.

## Examples

This example shows processing a return code and output parameter. Errorchecking code is removed to simplify this example.

// CREATE PROCEDURE TestParm @OutParm int OUTPUT AS

```
// SELECT au_lname FROM pubs.dbo.authors
```

```
// SELECT @OutParm = 88
```

### // RETURN 99

#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>

#define MAXBUFLEN 255

SQLHENV	henv = SQL_NULL_HENV;
SQLHDBC	hdbc1 = SQL_NULL_HDBC;
SQLHSTMT	hstmt1 = SQL_NULL_HSTMT;

```
int main() {
    RETCODE retcode;
    // SQLBindParameter variables.
    SWORD sParm1=0, sParm2=1;
    SDWORD cbParm1=SQL_NTS, cbParm2=SQL_NTS;
```

// Allocate statement handle.
retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt)

```
// Bind the return code to variable sParm1.
retcode = SQLBindParameter(hstmt1,1,SQL_PARAM_OUTPUT,SQ
   SQL INTEGER,0,0,&sParm1,0,&cbParm1);
// Bind the output parameter to variable sParm2.
retcode = SQLBindParameter(hstmt1,2,SQL_PARAM_OUTPUT,SQ
            SQL_INTEGER,0,0,&sParm2,0,&cbParm2);
// Execute the command.
retcode = SQLExecDirect(hstmt1, "{? = call TestParm(?)}", SQL_N<sup>r</sup>
// Show parameters are not filled.
printf("Before result sets cleared: RetCode = %d, OutParm = %d.\n",
   sParm1, sParm2);
// Clear any result sets generated.
while ((retcode = SQLMoreResults(hstmt1)) != SQL_NO_DATA)
 ;
// Show parameters are now filled.
printf("After result sets drained: RetCode = %d, OutParm = %d.\n",
```

```
sParm1, sParm2);
```

}

```
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
```

# Managing text and image Columns (ODBC)

The Microsoft® SQL Server<sup>™</sup> ODBC driver supports using **text** and **image** parameters and retrieving data from **text**, **ntext**, and **image** columns in result sets.

## How to use data-at-execution parameters (ODBC)

#### To use data-at-execution text, ntext, or image parameters

- 1. When calling **SQLBindParameter** to bind a program buffer to the statement parameter:
  - Use a *pcbValue* of SQL\_LEN\_DATA\_AT\_EXEC(*length*) where *length* is the total length of the **text**, **ntext**, or **image** parameter data in bytes.
  - Use an *rgbValue* of a program-defined parameter identifier.
- 2. Calling **SQLExecDirect** or **SQLExecute** returns SQL\_NEED\_DATA, which indicates that data-at-execution parameters are ready for processing.
- 3. For each data-at-execution parameter:
  - Call **SQLParamData** to get the program-defined parameter ID. It will return SQL\_NEED\_DATA if there is another data-at-execution parameter.
  - Call **SQLPutData** one or more times, to send the parameter data, until *length* is sent.
- 4. Call **SQLParamData** to indicate that all the data for the final data-atexecution parameter is sent. It will not return SQL\_NEED\_DATA.

## Examples

This example shows using **SQLPutData** to fill the data in a data-at-execution text parameter. Error-checking code is removed to simplify this example.

// Sample ODBC3 console application to write SQL\_LONGVARCHAD
// using SQLPutData.

// Assumes DSN has table:

// SQLSrvr: CREATE TABLE emp3 (NAME char(30), AGE int,

// BIRTHDAY datetime, Memo1 text)

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

#define TEXTSIZE 12000

SQLHENVhenv = SQL\_NULL\_HENV;SQLHDBChdbc1 = SQL\_NULL\_HDBC;SQLHSTMThstmt1 = SQL\_NULL\_HSTMT;

int main() {
 RETCODE retcode;

```
// SQLBindParameter variables.
SDWORD cbTextSize, lbytes;
//SQLParamData variable.
PTR pParmID;
//SQLPutData variables.
UCHAR Data[] =
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
    "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
```

```
"abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
"abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
"abcdefghijklmnopqrstuvwxyz";
```

SDWORD cbBatch = (SDWORD)sizeof(Data)-1;

"sa", SQL\_NTS, "MyPassWord", SQL\_NTS);

// Allocate statement handle.

retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt)

```
// Set parameters based on total data to send.
lbytes = (SDWORD)TEXTSIZE;
cbTextSize = SQL_LEN_DATA_AT_EXEC(lbytes);
// Bind the parameter marker.
retcode = SQLBindParameter(hstmt1, // hstmt
   1,
               // ipar
   SQL_PARAM_INPUT,
                            // fParamType
   SQL_C_CHAR, // fCType
   SQL_LONGVARCHAR,
                             // FSqlType
   lbytes,
                 // cbColDef
        // ibScale
   0,
   (VOID *)1,
                    // rgbValue
              // cbValueMax
   0.
   &cbTextSize);
                    // pcbValue
```

```
// Execute the command.
retcode = SQLExecDirect(hstmt1,
"INSERT INTO emp3 VALUES('Paul Borm', 46,'1950-11-24 00:00:(
          SQL_NTS);
// Check to see if NEED DATA; if yes, use SQLPutData.
retcode = SQLParamData(hstmt1, &pParmID);
if (retcode == SQL NEED DATA)
{
 while (lbytes > cbBatch)
 ł
   SQLPutData(hstmt1, Data, cbBatch);
   lbytes -= cbBatch;
 }
 // Put final batch.
 SQLPutData(hstmt1, Data, lbytes);
}
else
{
   ProcessErrorMessages(SQL_HANDLE_STMT, hstmt1,
          "SQLPutData Failed\n\n");
   return(9);
}
// Make final SQLParamData call.
retcode = SQLParamData(hstmt1, &pParmID);
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL HANDLE ENV, henv);
return(0);
```

}

## See Also

Bound vs. Unbound text and image Columns

**SQLBindParameter** 

Data-at-execution and text, ntext, or image Columns

<u>SQLPutData</u>

Managing text and image Columns

# How to use data-at-execution columns (ODBC)

#### To use data-at-execution text, ntext, or image columns

- 1. For each data-at-execution column, put special values into the buffers previously bound by **SQLBindCol**:
  - Into the *pcbValue* data value buffer, put SQL\_LEN\_DATA\_AT\_EXEC(*length*) where *length* is the total length of the **text**, **ntext**, or **image** column data in bytes.
  - Into the *rgbValue* data length buffer, put a program-defined column identifier.
- 2. Calling **SQLSetPos** returns SQL\_NEED\_DATA, which indicates that data-at-execution columns are ready for processing.
- 3. For each data-at-execution column:
  - Call **SQLParamData** to get the column array pointer. It will return SQL\_NEED\_DATA if there is another data-at-execution column.
  - Call **SQLPutData** one or more times to send the column data, until *length* is sent.
- 4. Call **SQLParamData** to indicate that all the data for the final data-atexecution column is sent. It will not return SQL\_NEED\_DATA.

## Examples

This example shows using **SQLGetData** to retrieve the data from a data-atexecution **text** column. Error-checking code was removed to simplify this example.

// Sample ODBC3 console application to read SQL\_LONGVARChar

// Assumes DSN has table: // SQLSrvr: CREATE TABLE emp3 (NAME char(30), AGE int, BIRTHDAY datetime, Memo1 text) // #include <stdio.h> #include <string.h> #include <windows.h> #include <sql.h> #include <sqlext.h> #include <odbcss.h> #define TEXTSIZE 12000 #define BUFFERSIZE 450 henv = SQL\_NULL\_HENV; SQLHENV SQLHDBC hdbc1 = SQL\_NULL\_HDBC; SQLHSTMT hstmt1 = SQL NULL HSTMT; int main() { **RETCODE** retcode; SWORD cntr; //SQLGetData variables. UCHAR Data[BUFFERSIZE]; SDWORD cbBatch = (SDWORD)sizeof(Data)-1; SDWORD cbTxtSize; // Clear data array. for(cntr = 0; cntr < BUFFERSIZE; cntr++)</pre> Data[cntr] = 0x00;// Allocate the ODBC environment and save handle.

// data using SQLGetData.

```
retcode = SQLAllocHandle (SQL_HANDLE_ENV, NULL, &henv);
// Notify ODBC that this is an ODBC 3.0 app.
retcode = SQLSetEnvAttr(henv, SQL ATTR ODBC VERSION,
         (SQLPOINTER) SQL_OV_ODBC3,
          SQL IS INTEGER);
// Allocate ODBC connection handle and connect.
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
retcode = SQLConnect(hdbc1, "MyDSN", SQL_NTS,
        "sa", SQL NTS, "MyPassWord, SQL NTS);
// Allocate statement handle; prepare, then execute command.
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt)
retcode = SQLExecDirect(hstmt1,
          "SELECT Memo1 FROM emp3",
          SQL_NTS);
// Get first row.
retcode = SQLFetch(hstmt1);
// Get the SQL LONG column.
cntr = 1;
do {
 retcode = SQLGetData(hstmt1, // hstmt
              // ipar
   1,
   SQL_C_CHAR,
                        // fCType
               // rgbValue
   Data,
   cbBatch,
                  // cbValueMax
   &cbTxtSize);
                     // pcbValue
 if (retcode != SQL_NO_DATA) {
 printf("GetData iteration %d, pcbValue = %d,\n",
    cntr++, cbTxtSize);
```

printf("Data = %s\n\n", Data);

}

```
} while (retcode != SQL_NO_DATA);
```

```
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
} // End Main.
```

## See Also

Bound vs. Unbound text and image Columns

**SQLBindCol** 

Data-at-execution and text/ntext/image Columns

**SQLPutData** 

Managing text and image Columns

# **Profiling ODBC Driver Performance (ODBC)**

The Microsoft® SQL Server<sup>™</sup> ODBC driver has two driver-specific options for profiling the performance of the driver.

The SQL Server ODBC driver can log performance statistics in file. The log file is a tab-delimited file that can be analyzed in any spreadsheet supporting tab-delimited files, such as Microsoft Excel.

The driver can also log long-running queries (queries that do not get a response from the server in a specified length of time). These queries can later be analyzed by programmers and database administrators.

## How to profile driver performance data (ODBC)

#### To log driver performance data using ODBC Administrator

- 1. In Control Panel, double-click **32-bit ODBC**.
- 2. Click the **User DSN**, **System DSN**, or **File DSN** tab.
- 3. Click the data source for which to log performance.
- 4. Click **Configure**.
- 5. Navigate the Microsoft SQL Server Configure DSN Wizard to the page with **Log ODBC driver statistics to the log file**.
- 6. Select **Log ODBC driver statistics to the log file**. In the box, place the name of the file where the statistics should be logged. Optionally, click **Browse** to browse the file system for the statistics log.

#### To log driver performance data programmatically

- Call SQLSetConnectAttr with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA\_LOG and *vParam* set to the full path and file name of the performance data log file. For example: "C:\\Odbcperf.log"
- 2. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA and *vParam* set to SQL\_PERF\_START to start logging performance data.
- 3. Optionally, call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_LOG\_NOW and *vParam* set to NULL to write a tab-delimited record of performance data to the performance data log file.

This can be done multiple times as the application runs.

4. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA and *vParam* set to SQL\_PERF\_STOP to stop logging performance data.

#### To pull driver performance data into an application

- 1. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA and *vParam* set to SQL\_PERF\_START to start profiling performance data.
- 2. Call **SQLGetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA and *pvParam* set to the address of a pointer to a SQLPERF structure. The first such call sets the pointer to the address of a valid SQLPERF structure that contains current performance data. The driver does not continually refresh the data in the performance structure. The application must repeat the call to **SQLGetConnectAttr** anytime it needs to refresh the structure with more current performance data.
- 3. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_DATA and *vParam* set to SQL\_PERF\_STOP to stop logging performance data.

The SQLPERF structure is defined in Odbcss.h as follows:

```
typedef struct sqlperf
{
   // Application profile statistics
   DWORD TimerResolution;
   DWORD SQLidu;
   DWORD SQLiduRows;
   DWORD SQLSelects;
   DWORD SQLSelectRows;
```

**DWORD** Transactions; **DWORD SQLPrepares; DWORD** ExecDirects; **DWORD SQLExecutes;** DWORD CursorOpens; **DWORD** CursorSize; **DWORD** CursorUsed: LDOUBLE PercentCursorUsed; LDOUBLE AvgFetchTime; LDOUBLE AvgCursorSize; LDOUBLE AvgCursorUsed; **DWORD SQLFetchTime;** DWORD SQLFetchCount; DWORD CurrentStmtCount; DWORD MaxOpenStmt; DWORD SumOpenStmt;

// Connection statistics
DWORD CurrentConnectionCount;
DWORD MaxConnectionsOpened;
DWORD SumConnectionsOpened;
DWORD SumConnectionTime;
LDOUBLE AvgTimeOpened;

// Network statistics
DWORD ServerRndTrips;
DWORD BuffersSent;
DWORD BuffersRec;
DWORD BytesSent;
DWORD BytesRec;

// Time statistics
DWORD msExecutionTime;

DWORD msNetworkServerTime;

# } SQLPERF;

## Examples

This example shows both the creation of a performance data log file and displaying performance data directly from the SQLPERF data structure. Error-checking code is removed to simplify this example.

// Sample showing the SQL Server ODBC driver-specific options
// to record performance statistics. The sample creates
// one file:
//
// C:\Odbcperf.log contains performance statistics and
// can be imported to a spreadsheet application such as Microsoft Excel
// as a tab-delimited file for analysis.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

SQLHENV	henv = SQL_NULL_HENV;
SQLHDBC	hdbc1 = SQL_NULL_HDBC;
SQLHSTMT	hstmt1 = SQL_NULL_HSTMT;

int main() {
 RETCODE retcode;
 // Pointer to the ODBC driver performance structure.
 SQLPERF \*PerfPtr;
 SQLINTEGER cbPerfPtr;

// Set options to log performance statistics.

```
// Allocate statement handle, then execute command.
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt1
retcode = SQLExecDirect(hstmt1,
```

```
"SELECT * FROM pubs.dbo.authors", SQL_NTS);
// Clear any result sets generated.
while ( ( retcode = SQLMoreResults(hstmt1) ) != SQL_NO_DATA )
.
```

```
retcode = SQLExecDirect(hstmt1,
    "SELECT * FROM pubs.dbo.stores", SQL_NTS);
// Clear any result sets generated.
while ( ( retcode = SQLMoreResults(hstmt1) ) != SQL_NO_DATA )
;
```

```
// Write current statistics to the performance log.
retcode = SQLSetConnectAttr
```

```
(hdbc1,
SQL_COPT_SS_PERF_DATA_LOG_NOW,
(SQLPOINTER)NULL,
SQL_IS_UINTEGER);
```

```
// Get pointer to current SQLPerf structure.
```

```
// Print a couple of statistics.
```

retcode = SQLGetConnectAttr

(hdbc1, SQL\_COPT\_SS\_PERF\_DATA,

(SQLPOINTER)&PerfPtr,

```
SQL_IS_POINTER,
```

&cbPerfPtr);

```
printf("SQLSelects = %d, SQLSelectRows = %d\n",
```

```
PerfPtr->SQLSelects, PerfPtr->SQLSelectRows);
```

/\* Clean up. \*/ SQLFreeHandle(SQL\_HANDLE\_STMT, hstmt1);

```
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
}
```

## See Also

Profiling ODBC Driver Performance

<u>SQLSetConnectAttr</u>

<u>SQLGetConnectAttr</u>

# How to log long-running queries (ODBC)

#### To log long-running queries using ODBC Administrator

- 1. In Control Panel, double-click **32-bit ODBC**.
- 2. Click the **User DSN**, **System DSN**, or **File DSN** tab.
- 3. Click the data source for which to log long-running queries.
- 4. Click **Configure**.
- 5. Navigate the Microsoft SQL Server Configure DSN Wizard to the page with **Save long-running queries to the log file**.
- 6. Select **Save long-running queries to the log file**. In the box, place the name of the file where the long-running queries should be logged. Optionally, click **Browse** to browse the file system for the query log.
- 7. Set a query time-out interval, in milliseconds, in the **Long query time** (milliseconds) box.

#### To log long-running queries data programmatically

- Call SQLSetConnectAttr with *fOption* set to SQL\_COPT\_SS\_PERF\_QUERY\_LOG and *vParam* set to the full path and file name of the long-running query log file. For example: C:\\Odbcqry.log
- 2. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_QUERY\_INTERVAL and *vParam* set to the time-out interval, in milliseconds.

- 3. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_QUERY and *vParam* set to SQL\_PERF\_START to start logging long-running queries.
- 4. Call **SQLSetConnectAttr** with *fOption* set to SQL\_COPT\_SS\_PERF\_QUERY and *vParam* set to SQL\_PERF\_STOP to stop logging long-running queries.

## Examples

This example shows the creation of a long-running query log file. Errorchecking code is removed to simplify this example.

// Sample showing the SQL Server ODBC driver-specific options
// to log long-running queries. Creates C:\Odbcqry.log, which
// contains a list of queries whose execution
// exceeds an interval set by the application.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <sqlext.h>
#include <odbcss.h>
```

SQLHENV	henv = SQL_NULL_HENV;
SQLHDBC	hdbc1 = SQL_NULL_HDBC;
SQLHSTMT	hstmt1 = SQL_NULL_HSTMT;

int main() {

**RETCODE** retcode;

// Allocate the ODBC environment and save handle.

```
retcode = SQLAllocHandle (SQL_HANDLE_ENV, NULL, &henv);
// Notify ODBC that this is an ODBC 3.0 app.
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
         (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_INTEGER
// Allocate ODBC connection handle and connect.
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
retcode = SQLConnect(hdbc1, "MyDSN", SQL_NTS,
      "sa", SQL_NTS, "MyPassWord", SQL_NTS));
// Set options to log long-running queries, including the
// file to use for the log.
retcode = SQLSetConnectAttr
         (hdbc1,
         SQL_COPT_SS_PERF_QUERY_LOG,
         &"c:\\odbcqry.log",
         SQL_NTS);
// Set the long-running query interval (in
// milliseconds). Note that for version 2.50 and 2.65
// drivers, this value is specified in seconds, not milliseconds.
retcode = SQLSetConnectAttr
         (hdbc1,
         SQL COPT SS PERF QUERY INTERVAL,
         (SQLPOINTER)3000,
         SQL IS UINTEGER);
// Start the long-running query log.
retcode = SQLSetConnectAttr
         (hdbc1,
         SQL_COPT_SS_PERF_QUERY,
         (SQLPOINTER)SQL_PERF_START,
         SQL_IS_UINTEGER);
```

// Allocate statement handle then execute commands.
retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc1, &hstmt2

```
retcode = SQLExecDirect(hstmt1,
    "SELECT * FROM pubs.dbo.authors", SQL_NTS);
// Clear any result sets generated.
while ( ( retcode = SQLMoreResults(hstmt1) ) != SQL_NO_DATA )
;
```

```
retcode = SQLExecDirect(hstmt1,
```

```
"SELECT * FROM pubs.dbo.stores", SQL_NTS);
// Clear any result sets generated.
while ( ( retcode = SQLMoreResults(hstmt1) ) != SQL_NO_DATA )
;
```

```
/* Clean up. */
SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
}
```

## See Also

Profiling ODBC Driver Performance

<u>SQLSetConnectAttr</u>

<u>SQLGetConnectAttr</u>

# How to process ODBC errors (ODBC)

Two ODBC function calls can be used to retrieve ODBC messages: **SQLGetDiagRec** and **SQLGetDiagField**. To obtain primary ODBC-related information in the *SQLState*, *pfNative*, and *ErrorMessage* diagnostic fields, call **SQLGetDiagRec** until it returns SQL\_NO\_DATA. For each diagnostic record, **SQLGetDiagField** can be called to retrieve individual fields. All driver-specific fields must be retrieved using **SQLGetDiagField**.

**SQLGetDiagRec** and **SQLGetDiagField** are processed by ODBC Driver Manager, not an individual driver. ODBC Driver Manager does not cache driverspecific diagnostic fields until a successful connection has been made. Calling **SQLGetDiagField** for driver-specific diagnostic fields is not possible before a successful connection. This includes the ODBC connection commands, even if they return SQL\_SUCCESS\_WITH\_INFO. Driver-specific diagnostic fields will not be available until the next ODBC function call.

## Examples

The following example shows a simple error handler that calls **SQLGetDiagRec** for the standard ODBC information. It then tests for a valid connection, and if there is, it calls **SQLGetDiagField** for the Microsoft® SQL Server<sup>™</sup> ODBC driver-specific diagnostic fields.

```
// Example of SQL Server ODBC driver-specific options
// on SQLGetDiagField.
//
// This application assumes the existence of the following
// stored procedure:
//
// CREATE PROCEDURE BadOne AS SELECT * FROM NotThere
//
// where no object named NotThere exists.
```

#include <stdio.h>

```
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

```
#define MAXBUFLEN 256
```

```
SQLHENVhenv = SQL_NULL_HENV;SQLHDBChdbc1 = SQL_NULL_HDBC;SQLHSTMThstmt1 = SQL_NULL_HSTMT;charlogstring[MAXBUFLEN] = "";
```

void ProcessLogMessages(SQLSMALLINT plm\_handle\_type, SQLHANDLE plm\_handle, char \*logstring, int ConnInd);

```
int main() {
```

```
RETCODE retcode;
```

```
// Allocate the ODBC environment and save handle.
retcode = SQLAllocHandle (SQL_HANDLE_ENV, NULL, &henv);
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
    (retcode != SQL_SUCCESS)) {
    printf("SQLAllocHandle(Env) Failed\n\n");
    return(9);
}
// Notify ODBC that this is an ODBC 3.0 app.
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
```

```
(SQLPOINTER) SQL_OV_ODBC3, SQL_IS_INTEGER
if( (retcode != SQL_SUCCESS_WITH_INFO) &&
```

```
(retcode != SQL_SUCCESS)) {
```

```
printf("SQLSetEnvAttr(ODBC version) Failed\n\n");
return(9);
```

```
}
```

```
// Allocate ODBC connection handle and connect.
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
if( (retcode != SQL SUCCESS WITH INFO) &&
  (retcode != SQL SUCCESS)) {
 printf("SQLAllocHandle(hdbc1) Failed\n\n");
 return(9);
}
retcode = SQLConnect(hdbc1, "MyDSN", SQL_NTS,
 "sa", SQL_NTS, "MyPassWord", SQL_NTS);
if ( (retcode != SQL_SUCCESS) &&
  (retcode != SQL SUCCESS WITH INFO)) {
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
      "SQLConnect() Failed\n\n", FALSE);
   return(9);
}
else {
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
         "\nConnect Successful\n\n", FALSE);
 }
// Allocate statement handle, and then execute command.
retcode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc1, &hstmt)
if ( (retcode != SQL_SUCCESS) &&
  (retcode != SQL_SUCCESS_WITH_INFO) ) {
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
         "SQLAllocHandle(hstmt1) Failed\n\n",
         TRUE);
   return(9);
}
```

```
retcode = SQLExecDirect(hstmt1, "exec BadOne", SQL_NTS);
 if ( (retcode != SQL_SUCCESS) &&
    (retcode != SQL SUCCESS WITH INFO)) {
    ProcessLogMessages(SQL_HANDLE_STMT, hstmt1,
         "SQLExecute() Failed\n\n", TRUE);
    return(9);
 }
 // Clear any result sets generated.
 while ((retcode = SQLMoreResults(hstmt1))!= SQL NO DATA)
   ;
 /* Clean up. */
 SQLFreeHandle(SQL_HANDLE_STMT, hstmt1);
 SQLDisconnect(hdbc1);
 SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
 SQLFreeHandle(SQL HANDLE ENV, henv);
 return(0);
}
void ProcessLogMessages(SQLSMALLINT plm_handle_type,
         SQLHANDLE plm_handle,
         char *logstring, int ConnInd)
{
 RETCODE
              plm_retcode = SQL_SUCCESS;
           plm_szSqlState[MAXBUFLEN] = "",
 UCHAR
      plm_szErrorMsg[MAXBUFLEN] = "";
             plm_pfNativeError = 0L;
 SDWORD
            plm_pcbErrorMsg = 0;
 SWORD
 SQLSMALLINT plm_cRecNmbr = 1;
 SDWORD
             plm_SS_MsgState = 0, plm_SS_Severity = 0;
 SQLINTEGER plm_Rownumber = 0;
            plm_SS_Line;
 USHORT
 SQLSMALLINT plm_cbSS_Procname, plm_cbSS_Srvname;
```

SQLCHAR plm\_SS\_Procname[MAXNAME], plm\_SS\_Srvname

printf(logstring);

```
while (plm retcode != SQL NO DATA FOUND) {
 plm_retcode = SQLGetDiagRec(plm_handle_type, plm_handle,
   plm_cRecNmbr, plm_szSqlState, &plm_pfNativeError,
   plm_szErrorMsg, MAXBUFLEN - 1, &plm_pcbErrorMsg);
 // Note that if the application has not yet made a
 // successful connection, the SQLGetDiagField
 // information has not yet been cached by ODBC
 // Driver Manager and these calls to SQLGetDiagField
 // will fail.
 if (plm_retcode != SQL_NO_DATA_FOUND) {
   if (ConnInd) {
    plm_retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
      SQL_DIAG_ROW_NUMBER, &plm_Rownumber,
      SQL IS INTEGER,
      NULL);
    plm retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
      SQL_DIAG_SS_LINE, &plm_SS_Line,
      SQL_IS_INTEGER,
      NULL);
    plm_retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
      SQL_DIAG_SS_MSGSTATE, &plm_SS_MsgState,
      SQL IS INTEGER,
      NULL):
    plm_retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
```

```
SQL_DIAG_SS_SEVERITY, &plm_SS_Severity,
      SQL_IS_INTEGER,
      NULL);
    plm_retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
      SQL_DIAG_SS_PROCNAME, &plm_SS_Procname,
      sizeof(plm_SS_Procname),
      &plm_cbSS_Procname);
    plm retcode = SQLGetDiagField(
      plm_handle_type, plm_handle, plm_cRecNmbr,
      SQL_DIAG_SS_SRVNAME, &plm_SS_Srvname,
      sizeof(plm_SS_Srvname),
      &plm_cbSS_Srvname);
   }
  printf("szSqlState = %s\n",plm_szSqlState);
   printf("pfNativeError = %d\n",plm_pfNativeError);
   printf("szErrorMsg = %s\n",plm_szErrorMsg);
   printf("pcbErrorMsg = %d\n\n",plm_pcbErrorMsg);
   if (ConnInd) {
    printf("ODBCRowNumber = %d\n", plm Rownumber);
    printf("SSrvrLine = %d\n", plm_Rownumber);
    printf("SSrvrMsgState = %d\n",plm SS MsgState);
    printf("SSrvrSeverity = %d\n",plm_SS_Severity);
    printf("SSrvrProcname = %s\n",plm_SS_Procname);
    printf("SSrvrSrvname = %s\n\n",plm_SS_Srvname);
   }
 }
 plm_cRecNmbr++; //Increment to next diagnostic record.
} // End while.
```

See Also

}

Handling Errors and MessagesSQLGetDiagFieldDiagnostic Records and Fields

# How to bulk copy with the SQL Server ODBC driver (ODBC)

When used with Microsoft<sup>®</sup> SQL Server<sup>™</sup> version 7.0, the SQL Server ODBC driver supports the same bulk copy functions supported by the DB-Library API.

# How to bulk copy without a format file (ODBC)

#### To bulk copy without a format file

- 1. Allocate an environment handle and a connection handle.
- 2. Set SQL\_COPT\_SS\_BCP and SQL\_BCP\_ON to enable bulk copy operations.
- 3. Connect to Microsoft<sup>®</sup> SQL Server<sup>™</sup>.
- 4. Call **bcp\_init** to set the following information:
  - The name of the table or view to bulk copy from or to.
  - The name of the data file that contains the data to copy into the database or that receives data when copying from the database.
  - The name of a data file to receive any bulk copy error messages (specify NULL if you do not want a message file).
  - The direction of the copy: DB\_IN from the file to the view or table, or DB\_OUT to the file from the table or view.
- 5. Call **bcp\_exec** to execute the bulk copy operation.

When DB\_OUT is set with these steps, the file is created in native format. The file can then be bulk copied into a server by following these same steps, except that DB\_OUT is set instead of DB\_IN. This works only if both the source and target tables have exactly the same structure.

## Examples

The following example shows using bulk copy functions to create a native mode data file. Most error-checking code was removed to simplify this example.

// Sample showing ODBC BCP\_OUT in native mode format.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

SQLHENV henv = SQL\_NULL\_HENV; HDBC hdbc1 = SQL\_NULL\_HDBC;

```
int main() {
    RETCODE retcode;
```

```
// Bulk copy variables.
SDWORD cRows;
```

```
// Allocate the ODBC environment and save handle.
retcode = SQLAllocHandle (SQL_HANDLE_ENV, NULL, &henv);
```

```
// Notify ODBC that this is an ODBC 3.0 app.
retcode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
        (SQLPOINTER) SQL_OV_ODBC3,
        SQL_IS_INTEGER);
```

```
// Allocate ODBC connection handle, set bulk copy mode, and
// then connect.
```

```
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
retcode = SQLSetConnectAttr(hdbc1, SQL_COPT_SS_BCP,
```

```
(void *)SQL_BCP_ON,
            SQL IS INTEGER);
retcode = SQLConnect(hdbc1, "MyDSN", SQL_NTS,
     "sa", SQL_NTS, "MyPassWord", SQL_NTS);
// Initialize the bulk copy.
retcode = bcp_init(hdbc1, "pubs..authors", "c:\\BCPODBC.bcp",
        "c:\\BCPERROR.out, DB_OUT);
// Note that the test is for the bulk copy return of SUCCEED,
// not the ODBC return of SQL_SUCCESS.
if ( (retcode != SUCCEED) )
{
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
          "bcp_init(hdbc1) Failed\n\n");
   return(9);
}
// Execute the bulk copy.
retcode = bcp_exec(hdbc1, &cRows);
if ( (retcode != SUCCEED) )
{
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
          "bcp_exec(hdbc1) Failed\n\n");
   return(9);
}
printf("Number of rows bulk copied out = %d.\n", cRows);
/* Clean up. */
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL HANDLE ENV, henv);
return(0);
```

}

The data file created by this sample is a native mode file. To bulk copy the data back into the table, recompile the application after changing the **bcp\_init** call from BCP\_OUT to BCP\_IN. To use the file as native-mode input to the **bcp** utility, enter at a command prompt:

bcp *MyDB..DateTable* in c:\BCPODBC.bcp /n /S*MyServer* /Usa /P*MyPassWord* 

## See Also

<u>bcp\_exec</u> <u>Using Data Files and Format Files</u> <u>bcp\_init</u>

# How to bulk copy a SELECT result set (ODBC)

#### To bulk copy out the result set of a SELECT statement

- 1. Allocate an environment handle and a connection handle.
- 2. Set SQL\_COPT\_SS\_BCP and SQL\_BCP\_ON to enable bulk copy operations.
- 3. Connect to Microsoft<sup>®</sup> SQL Server<sup>™</sup>.
- 4. Call **bcp\_init** to set the following information:
  - Specify NULL for the *szTable* parameter.
  - The name of the data file that receives result set data.
  - The name of a data file to receive any bulk copy error messages (specify NULL if you do not want a message file).
  - The direction of the copy: DB\_OUT.
- 5. Call **bcp\_control**, set *eOption* to BCPHINTS and place in *iValue* a pointer to a SQLTCHAR array containing the SELECT statement.
- 6. Call **bcp\_exec** to execute the bulk copy operation.

When using these steps the file is created in native format. You can convert the data values to other data types by using bcp\_colfmt, for more information, see the How to create a bulk copy format file (ODBC) section.

### Examples

The following example shows using bulk copy functions to bulk copy out the result set of a SELECT statement. Most error-checking code is removed to simplify this example.

// Sample showing ODBC BCP\_OUT of a SELECT result set // using native mode format.

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

```
SQLHENVhenv = SQL_NULL_HENV;HDBChdbc1 = SQL_NULL_HDBC;
```

int main() {
 RETCODE retcode;

```
// Bulk copy variables.
SDWORD cRows;
SQLTCHAR szBCPQuery[] =
   "SELECT LastName, FirstName FROM Northwind.dbo.Employee
```

```
// Allocate the ODBC environment and save handle.
retcode = SQLAllocHandle (SQL_HANDLE_ENV, NULL, &henv);
```

// Allocate ODBC connection handle, set bulk copy mode, and

```
// then connect.
retcode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc1);
retcode = SQLSetConnectAttr(hdbc1, SQL COPT SS BCP,
            (void *)SQL_BCP_ON,
            SQL IS INTEGER);
retcode = SQLConnect(hdbc1, "MyDSN", SQL_NTS,
     "sa", SQL NTS, "MyPassWord", SQL NTS);
// Initialize the bulk copy.
retcode = bcp_init(hdbc1, NULL, "c:\\BCPODBC.bcp",
        "c:\\BCPERROR.out, DB OUT);
// Note that the test is for the bulk copy return of SUCCEED,
// not the ODBC return of SQL_SUCCESS.
if ( (retcode != SUCCEED) )
{
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
          "bcp_init(hdbc1) Failed\n\n");
   return(9);
}
// Specify the query to use.
retcode = bcp_control(hdbc1, BCPHINTS, (void *)szBCPQuery);
if ( (retcode != SUCCEED) )
{
   ProcessLogMessages(SQL_HANDLE_DBC, hdbc1,
          "bcp_control(hdbc1) Failed\n\n");
   return(9);
}
// Execute the bulk copy.
retcode = bcp_exec(hdbc1, &cRows);
if ( (retcode != SUCCEED) )
{
```

```
}
```

## See Also

bcp\_init

bcp\_control

<u>bcp\_exec</u>

# How to create a bulk copy format file (ODBC)

#### To create a bulk copy format file

- 1. Allocate an environment handle and a connection handle.
- 2. Set SQL\_COPT\_SS\_BCP and SQL\_BCP\_ON to enable bulk copy operations.
- 3. Connect to Microsoft<sup>®</sup> SQL Server<sup>™</sup>.
- 4. Call **bcp\_init** to set the following information:
  - The name of the table or view to bulk copy from or to.
  - The name of the data file that contains the data to copy into the database or that receives data when copying from the database.
  - The name of a data file to receive any bulk copy error messages (specify NULL if you do not want a message file).
  - The direction of the copy: DB\_OUT to the file from the table or view.
- 5. Call **bcp\_columns** to set the number of columns.
- 6. Call **bcp\_colfmt** for each column to define its characteristics in the data file.
- 7. Call **bcp\_writefmt** to create a format file describing the data file to be created by the bulk copy operation.

8. Call **bcp\_exec** to execute the bulk copy operation.

A bulk copy operation run in this way creates both a data file containing the bulk copied data and a format file describing the layout of the data file.

## Examples

The following example shows using bulk copy functions to create both a data file and a format file. Error-checking code was removed to simplify this example.

```
// Sample showing ODBC BCP_OUT creating a format file.
//
// Assumes server has:
// CREATE TABLE BCPDate (cola int, colb datetime)
```

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <sqlext.h>
#include <odbcss.h>
```

SQLHENV henv = SQL\_NULL\_HENV; HDBC hdbc1 = SQL\_NULL\_HDBC;

int main() {
 RETCODE retcode;
 // BCP variables.
 SDWORD cRows;

// Allocate the ODBC environment and save handle.
retcode = SQLAllocHandle (SQL\_HANDLE\_ENV, NULL, &henv);

```
// Initialize the bulk copy.
```

```
retcode = bcp_init(hdbc1, "pubs..BCPDate", "c:\\BCPODBC.bcp",
NULL, DB_OUT);
```

```
// Set the number of output columns.
```

```
retcode = bcp_columns(hdbc1, 2);
```

// Describe the format of column 1 in the data file.
retcode = bcp\_colfmt(hdbc1, 1, SQLCHARACTER, -1, 5, NULL, 0,
// Describe the format of column 2 in the data file.
retcode = bcp\_colfmt(hdbc1, 2, SQLCHARACTER, -1, 20, NULL, (

```
// Create the format file.
retcode = bcp_writefmt(hdbc1, "c:\\BCPFMT.fmt");
```

```
// Execute the bulk copy.
retcode = bcp_exec(hdbc1, &cRows);
```

printf("Number of rows bulk copied out = %d.\n", cRows);

```
/* Clean up. */
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
```

```
return(0);
}
```

To bulk copy this data file back into the database, see How to bulk copy using a format file. To use this data file as the input to the **bcp** utility, enter at a command prompt:

bcp pubs..BCPDate in C:\Bcpodbc.bcp /fc:\Bcpfmt.fmt /SMyServer /Usa /PMyPassWord

## See Also

<u>bcp\_colfmt</u>

bcp\_writefmt

bcp\_columns

How to bulk copy by using a format file (ODBC)

<u>bcp\_exec</u>

**Using Data Files and Format Files** 

bcp\_init

# How to bulk copy by using a format file (ODBC)

#### To bulk copy by using a format file

- 1. Allocate an environment handle and a connection handle.
- 2. Set SQL\_COPT\_SS\_BCP and SQL\_BCP\_ON to enable bulk copy operations.
- 3. Connect to Microsoft<sup>®</sup> SQL Server<sup>™</sup>.
- 4. Call **bcp\_init** to set the following information:
  - The name of the table or view to bulk copy from or to.
  - The name of the data file that contains the data to copy into the database or that receives data when copying from the database.
  - The name of a data file to receive any bulk copy error messages (specify NULL if you do not want a message file).
  - The direction of the copy: DB\_IN from the file to the table or view.
- 5. Call **bcp\_readfmt** to read the format file describing the data file to be used by the bulk copy operation.
- 6. Call **bcp\_exec** to execute the bulk copy operation.

### Examples

The following example shows using bulk copy functions with both an data file

and format file. Error-checking code was removed to simplify this example.

// Sample showing ODBC BCP\_IN using a format file.
//

// Assumes server has:

// CREATE TABLE BCPDate (cola int, colb datetime)

// Assumes you have the format file and datafile from the example

// in How to create a bulk copy format file.

#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <sqlext.h>
#include <odbcss.h>

SQLHENVhenv = SQL\_NULL\_HENV;HDBChdbc1 = SQL\_NULL\_HDBC;

```
int main() {
    RETCODE retcode;
    // BCP variables.
    SDWORD cRows;
```

"sa", SQL\_NTS, "MyPassWord", SQL\_NTS);

// Initialize the bulk copy.

retcode = bcp\_init(hdbc1, "pubs..BCPDate", "c:\\BCPODBC.bcp", NULL, DB\_IN);

```
// Read the format file.
retcode = bcp_readfmt(hdbc1, "c:\\BCPFMT.fmt");
```

```
// Execute the bulk copy.
retcode = bcp_exec(hdbc1, &cRows);
```

printf("Number of rows bulk copied in = %d.\n", cRows);

```
/* Clean up. */
SQLDisconnect(hdbc1);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
}
```

## See Also

bcp\_exec How to bulk copy by using a format file bcp\_init Using Data Files and Format Files bcp\_readfmt

# How to bulk copy data from program variables (ODBC)

#### To use bulk copy functions directly on program variables

- 1. Allocate an environment handle and a connection handle.
- 2. Set SQL\_COPT\_SS\_BCP and SQL\_BCP\_ON to enable bulk copy operations.
- 3. Connect to Microsoft<sup>®</sup> SQL Server<sup>™</sup>.
- 4. Call **bcp\_init** to set the following information:
  - The name of the table or view to bulk copy from or to.
  - Specify NULL for the name of the data file.
  - The name of an data file to receive any bulk copy error messages (specify NULL if you do not want a message file).
  - The direction of the copy: DB\_IN from the application to the view or table or DB\_OUT to the application from the table or view.
- 5. Call **bcp\_bind** for each column in the bulk copy to bind the column to a program variable.
- 6. Fill the program variables with data, and call **bcp\_sendrow** to send a row of data.
- 7. After several rows have been sent, call **bcp\_batch** to checkpoint the

rows already sent. It is good practice to call **bcp\_batch** at least once per 1000 rows.

8. After all rows have been sent, call **bcp\_done** to complete the operation.

You can vary the location and length of program variables during a bulk copy operation by calling **bcp\_colptr** and **bcp\_collen**.

Use **bcp\_control** to set various bulk copy options. Use **bcp\_moretext** to send **text**, **ntext**, and **image** data in segments to the server.

## Examples

The following example shows using bulk copy functions to bulk copy data from program variables to SQL Server using **bcp\_bind** and **bcp\_sendrow**. Error-checking code is removed to simplify this example.

```
// Sample showing ODBC bulk copy from program variables
// bound with bcp bind; data sent with bcp sendrow.
||
// Assumes server has:
||
// CREATE TABLE BCPSource (cola int PRIMARY KEY,
            colb CHAR(10) NULL)
//
// CREATE TABLE BCPTarget (cola int PRIMARY KEY,
//
             colb CHAR(10) NULL)
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>
```

```
SQLHENVhenv = SQL_NULL_HENV;HDBChdbc1 = SQL_NULL_HDBC, hdbc2 = SQL_NULL_HDE
```

SQLHSTMT hstmt2 = SQL\_NULL\_HSTMT;

int main() {
 RETCODE retcode;

// BCP variables. char \*terminator = "\0"; // bcp\_done takes a different format return code // because it returns number of rows bulk copied // after the last bcp\_batch call. DBINT cRowsDone; // Set up separate return code for bcp\_sendrow so // it is not using the same retcode as SQLFetch. RETCODE SendRet;

// Column variables. // cbCola and cbColb must be defined right before // Cola and szColb because they are used as // bulk copy indicator variables. struct ColaData{ SQLINTEGER cbCola; SQLINTEGER Cola; } ColaInst; struct ColbData{ SQLINTEGER cbColb; SQLCHAR szColb[11]; } ColbInst;

// Initialize the bulk copy.

retcode = bcp\_init(hdbc1, "pubs..BCPTarget", NULL, NULL, DB\_IN);

// Bind the program variables for the bulk copy.

retcode = bcp\_bind(hdbc1, (BYTE \*)&ColaInst.cbCola, 4,

SQL\_VARLEN\_DATA, NULL, (INT)NULL, SQLINT4, 1);

// Could normally use strlen to calculate the bcp\_bind

// cbTerm parameter, but this terminator is a null byte

// (\0), which gives strlen a value of 0. Explicitly give // cbTerm a value of 1.

```
retcode = bcp_bind(hdbc1, (BYTE *)&ColbInst.cbColb, 4, 11,
terminator, 1, SQLCHARACTER, 2);
```

// Allocate second ODBC connection handle so that bulk copy
// and cursor operations do not conflict.

retcode = SQLAllocHandle(SQL\_HANDLE\_DBC, henv, &hdbc2); retcode = SQLConnect(hdbc2, "MyDSN", SQL\_NTS,

"sa", SQL\_NTS, "MyPassWord", SQL\_NTS);

// Allocate ODBC statement handle.

retcode = SQLAllocHandle(SQL\_HANDLE\_STMT, hdbc2, &hstmt.

// Bind the SELECT statement to the same program variables
// bound to the bulk copy operation.

retcode = SQLBindCol(hstmt2, 1, SQL\_C\_SLONG, &ColaInst.Cola

```
&ColaInst.cbCola);
retcode = SQLBindCol(hstmt2, 2, SQL_C_CHAR, &ColbInst.szColl
           &ColbInst.cbColb);
// Execute a SELECT statement to build a cursor containing
// the data to be bulk copied to the new table.
retcode = SQLExecDirect(hstmt2,
           "SELECT * FROM BCPSource",
           SQL_NTS);
// Go into a loop fetching rows from the cursor until
// each row is fetched. Because the bcp_bind calls
// and SQLBindCol calls each reference the same
// variables, each fetch fills the variables used by
// bcp_sendrow, so all you have to do to send the data
// to SQL Server is to call bcp_sendrow.
while ((retcode = SQLFetch(hstmt2)) != SQL_NO_DATA) {
 if ( (retcode != SQL_SUCCESS) &&
   (retcode != SQL SUCCESS WITH INFO)) {
     // Process error.
     return(9);
  }
 if ( (SendRet = bcp_sendrow(hdbc1) ) != SUCCEED ) {
   // Process error.
   return(9);
 }
}
// Signal the end of the bulk copy operation.
cRowsDone = bcp_done(hdbc1);
printf("Number of rows bulk copied after last bcp batch
       call = %d.\n", cRowsDone);
/* Clean up. */
```

```
SQLFreeHandle(SQL_HANDLE_STMT, hstmt2);
SQLDisconnect(hdbc1);
```

```
SQLFreeHandle(SQL_HANDLE_DBC, hdbc1);
SQLDisconnect(hdbc2);
SQLFreeHandle(SQL_HANDLE_DBC, hdbc2);
SQLFreeHandle(SQL_HANDLE_ENV, henv);
return(0);
```

## See Also

bcp\_batch

bcp\_init

bcp\_bind

bcp\_sendrow

bcp\_done

Bulk Copying from Program Variables