SQL Packager 6 documentation

About SQL Packager

SQL Packager enables you to package the structure and contents of a Microsoft SQL Server database, database upgrade, or any SQL script, as a .NET executable file or a C# project.

For more information, see the SQL Packager product page.

Quick links

SQL Packager 6.4 release notes

Packaging a database as an .EXE

Packaging an upgrade as a C# project

Troubleshooting

Requirements

To use SQL Packager you need:

- One of the following Microsoft Windows operating systems:
  - Windows 2000
  - Windows XP
  - Windows Server 2003
  - Windows Vista
  - Windows 7
  - Windows Server 2008
  - Windows 8
- SQL Server client-side tools
- Microsoft .NET Framework 2.0 or later
- MDAC 2.8 or later
- 128 MB RAM
- 200 MB hard disk space

The following versions of SQL Server are supported:

- SQL Server 2005
- SQL Server 2008
- SQL Server 2008 R2
- SQL Server on Amazon RDS

Please note that this product has not been tested on an Itanium server.

Installing

Most Redgate products are available as part of a bundle. You can select which individual products to install when you run the installer.

When you install a non-free product, you have 14 days to evaluate the product. For the DLM Automation Suite, DLM Automation Suite for Oracle, SQL Source Control, Schema Compare for Oracle, Data Compare for Oracle, and Source Control for Oracle, you have 28 days. For more information, see Licensing.

To install a Redgate product:

1. Download the product from the website.
2. Run the installer and follow the instructions.

The product is listed on the **Start** menu under **Red Gate**.

Licensing

When you install most Redgate products (apart from free ones), you have **14 days** to evaluate them without purchase.

For a few products, you have 28 days: DLM Automation Suite, DLM Automation Suite for Oracle, SQL Prompt, SQL Source Control, Source Control for Oracle.

If you need more time to evaluate a product, email licensing@red-gate.com.

Finding your serial number

When you buy a license for a product, we'll send you an invoice that contains your serial number to activate the product. Your invoice shows how many instances of a product the serial number can be used to activate. For information about how to activate, see Activating.

If you can't find your invoice, you can view your serial numbers at red-gate.com/myserialnumbers. You'll need to log in to your Redgate account with the email address and password you provided when you bought the product.

> If you need to reinstall products on the same computer (eg after installing a new operating system), you can reactivate them using the same serial number. This doesn't affect the number of distinct activations for the serial number. For information about moving a serial number to a different computer, see below.

Serial numbers for bundles and suites

If you've bought a bundle or suite of products, your serial number activates all the products in the bundle or suite. For bundles containing both server and client tools (such as the SQL DBA Bundle) you will have two serial numbers.

If you deactivate a bundle or suite serial number, all products using that serial number will be deactivated.

For information on which products are included in a bundle, see Bundle history.

Changing the serial number used to activate a product

To change the serial number used to activate a product, on the **Help** menu, select **Enter Serial Number**. For some products, you will need to deactivate the old serial number first.

Moving a serial number to a different computer

To move a serial number to a different computer, deactivate the serial number on the old computer, then use it to activate the product on the new computer.

To deactivate a serial number, on the **Help** menu, select **Deactivate Serial Number**. If the Deactivate Serial Number menu item isn't available, use the deactivation tool.

If you can't deactivate a serial number, use the Request Extra Activations page to request more activations for your serial number. You'll need to provide your serial number and the reason for the additional activations.

Activating

This page applies to a number of Redgate products, so the screenshots below may not match your product.

When you activate a product with your serial number, the licensing and activation program sends an activation request to the Redgate activation server, using checksums of attributes from your computer. The checksums sent to the activation server do not contain any details that might pose a security risk. The activation server returns an activation response and an encrypted key to unlock the software. The licensing and activation program should activate your product within a few seconds.

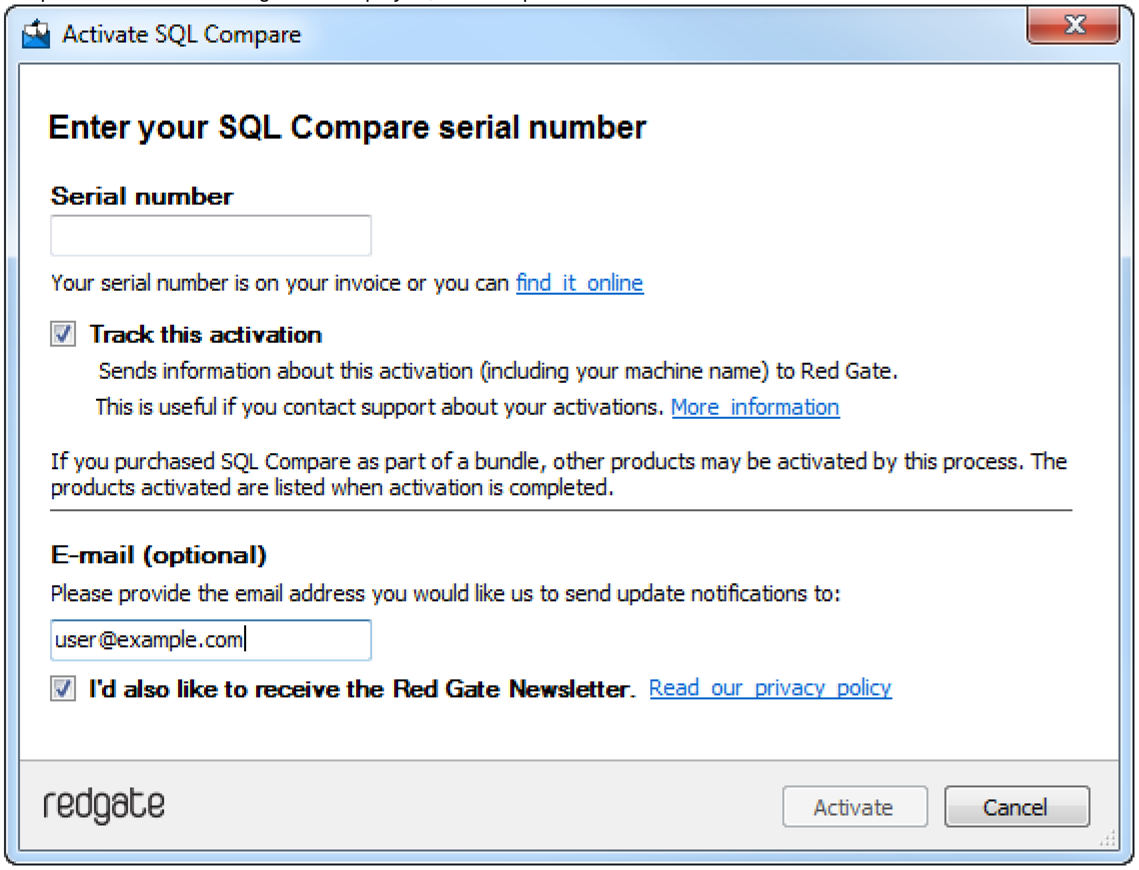
If you experience problems with activating your products, you'll be directed to activate manually.

- Activating using the GUI
- Activating using the command line
- Manual activation

Activating using the GUI

These instructions apply to a number of Redgate products, so the screenshots below may not match your product.

To activate your products:

1. On the **Help** menu, click **Enter Serial Number**.
   The product activation dialog box is displayed, for example:
   
   Activate SQL Compare

   Enter your SQL Compare serial number

   Serial number

   Your serial number is on your invoice or you can find it online

   ☑ Track this activation

   Sends information about this activation (including your machine name) to Red Gate.

   This is useful if you contact support about your activations. More information

   If you purchased SQL Compare as part of a bundle, other products may be activated by this process. The products activated are listed when activation is completed.

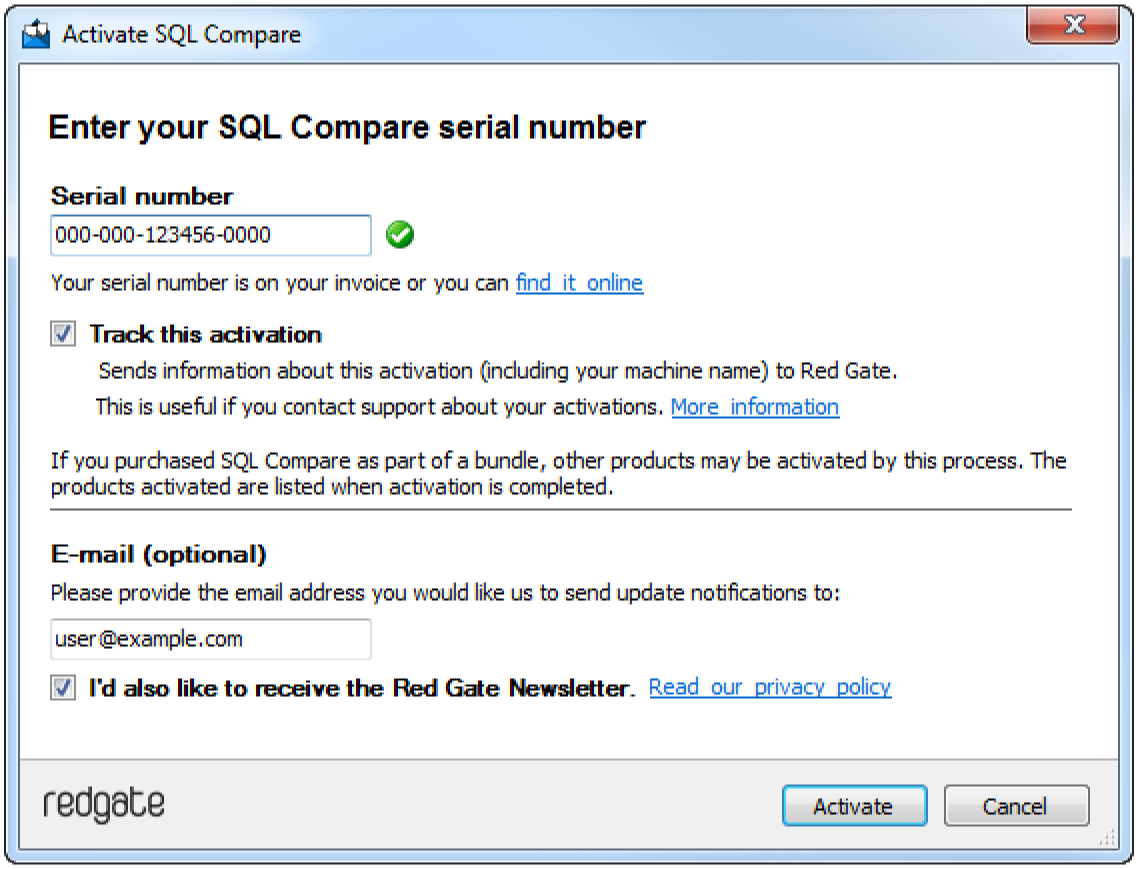
   E-mail (optional)

   Please provide the email address you would like us to send update notifications to:

   user@example.com

   ☑ I'd also like to receive the Red Gate Newsletter. Read our privacy policy

2. Enter your serial number.
   When you have entered a valid serial number,
   ✓
   is displayed next to the serial number box:


Activate SQL Compare

Enter your SQL Compare serial number

Serial number

000-000-123456-0000

Your serial number is on your invoice or you can find it online

[✓] Track this activation

Sends information about this activation (including your machine name) to Red Gate.

This is useful if you contact support about your activations. More information

If you purchased SQL Compare as part of a bundle, other products may be activated by this process. The products activated are listed when activation is completed.

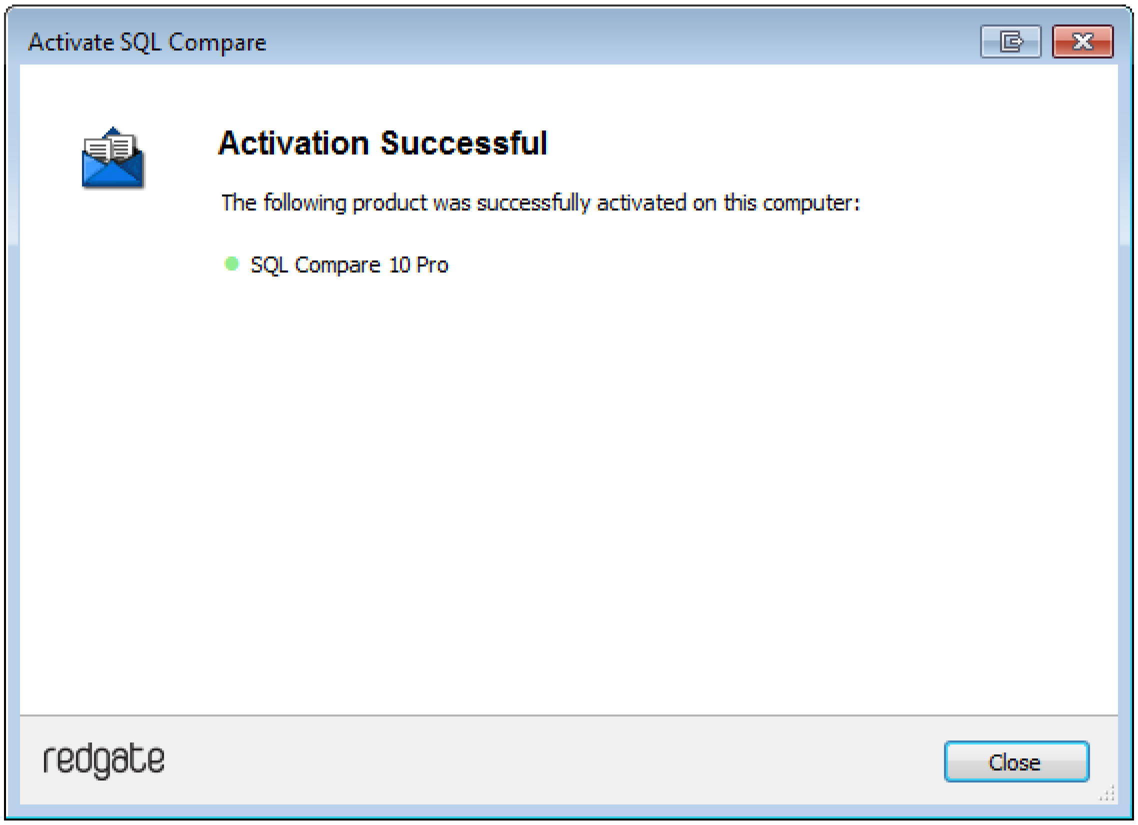
E-mail (optional)

Please provide the email address you would like us to send update notifications to:

user@example.com

[✓] I'd also like to receive the Red Gate Newsletter. Read our privacy policy

3. If you want to receive email updates from Redgate, enter your email address.
   The list of identifiers and your email address may already be populated using information available to the licensing client from the Windows installation on your computer. No information is sent back to Redgate when the fields are populated.
   When you activate your product, the optional information you entered is recorded by Redgate with your serial number. Your email address is not linked to the data collected should you consent to participate in the Quality Improvement Program provided with some Red Gate products.
4. Click **Activate**.
   Your activation request is sent to the Red Gate activation server.
   When your activation has been confirmed, the **Activation successful** page is displayed, for example:


Activate SQL Compare

Activation Successful

The following product was successfully activated on this computer:

● SQL Compare 10 Pro

If there is a problem with your activation request, an error dialog box is displayed. For information about activation errors and what you can do to resolve them, see Troubleshooting licensing and activation errors. Depending on the error, you may want to try manual activation.

5. Click **Close**.
   You can now continue to use your product.

Activating using the command line

Open a command prompt, navigate to the folder where your product executable file is located and run a command with the following syntax:

```
<name of productEXE> /activateSerial:<serialNumber>
```

For example:

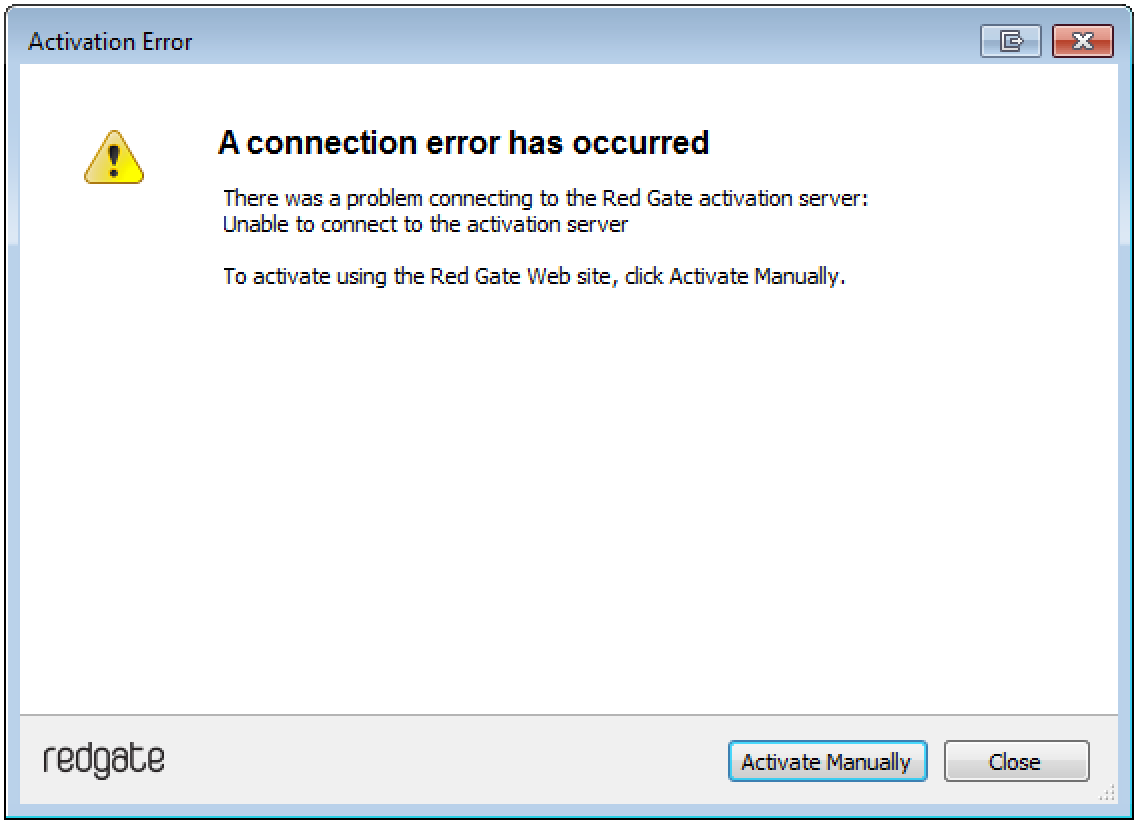
```
sqlcompare /activateSerial:123-456-789012-ABCD
```

The product activation dialog box is displayed. Follow the instructions below.

Manual activation

Manual activation enables you to activate products when your computer does not have an internet connection or your internet connection does not allow SOAP requests. You will need access to another computer that does have an internet connection.

You can use manual activation whenever the **Activation Error** dialog box is displayed and the **Activate Manually** button is available, for example:


Activation Error

A connection error has occurred

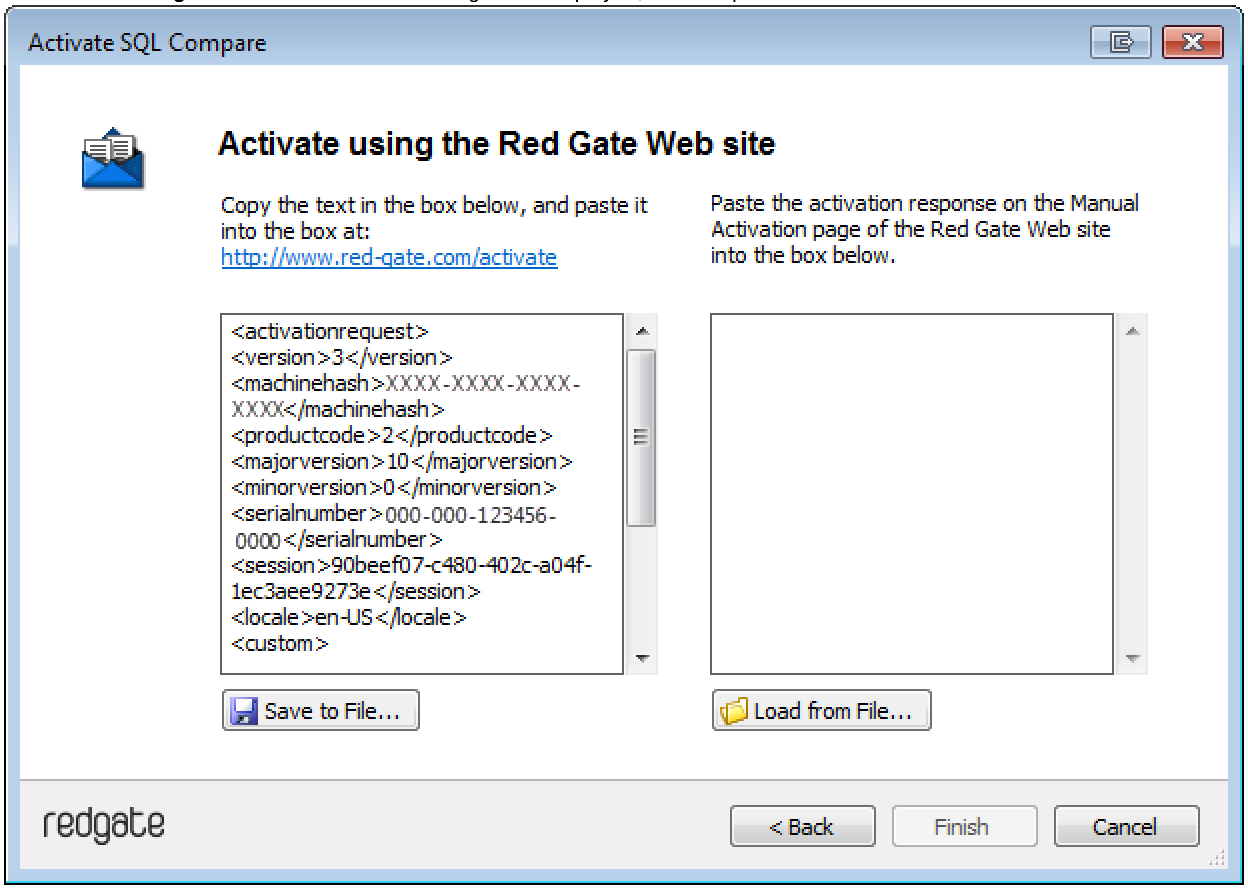
There was a problem connecting to the Red Gate activation server:
Unable to connect to the activation server

To activate using the Red Gate Web site, click Activate Manually.

To activate manually:

1. On the error dialog box, click **Activate Manually**.

   The **Activate using the Red Gate Web site** dialog box is displayed, for example:


Activate SQL Compare

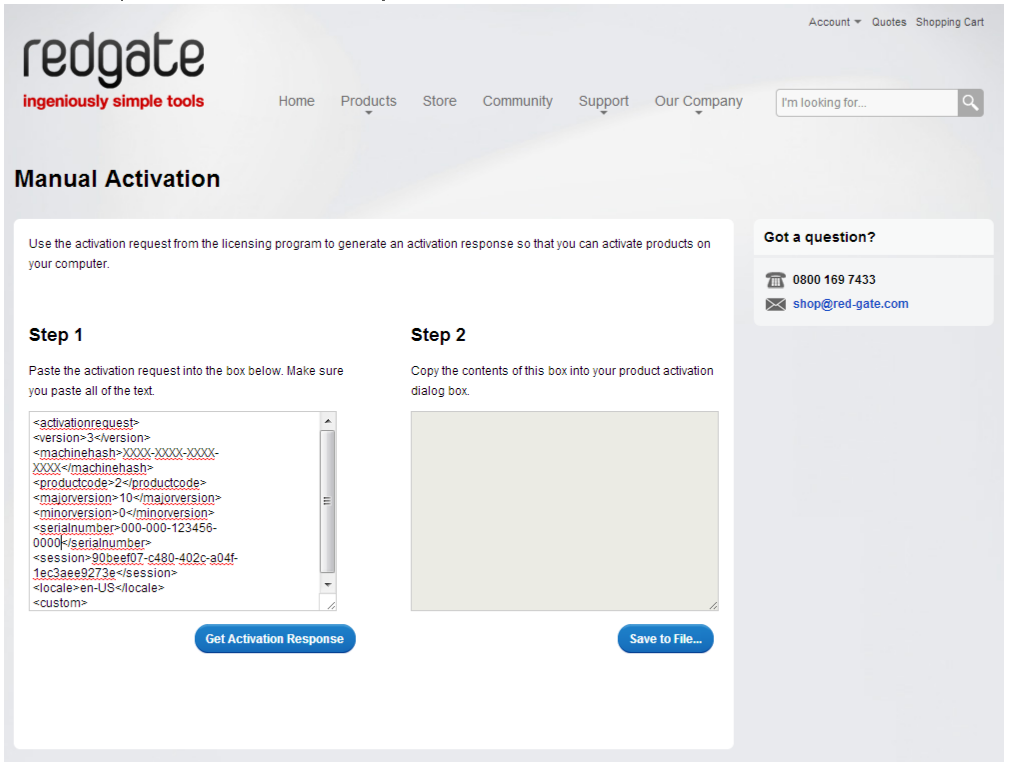
Activate using the Red Gate Web site

Copy the text in the box below, and paste it into the box at:
http://www.red-gate.com/activate

```
<activationrequest>
<version>3</version>
<machinehash>XXXX-XXXX-XXXX-
XXXX</machinehash>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<serialnumber>000-000-123456-
0000</serialnumber>
<session>90beef07-c480-402c-a04f-
1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

Paste the activation response on the Manual Activation page of the Red Gate Web site into the box below.

2. Copy all of the activation request, and **leave this dialog box open** (if you close the dialog box, you may have to start again).
   Alternatively you can save the activation request, for example to a location on your network or to a USB device.
3. On a computer that has an Internet connection, go to the **Manual Activation** page at http://www.red-gate.com/activate and paste the activation request into the box under **Step 1**.



Manual Activation

Use the activation request from the licensing program to generate an activation response so that you can activate products on your computer.

Got a question?

0800 169 7433

shop@red-gate.com

Step 1

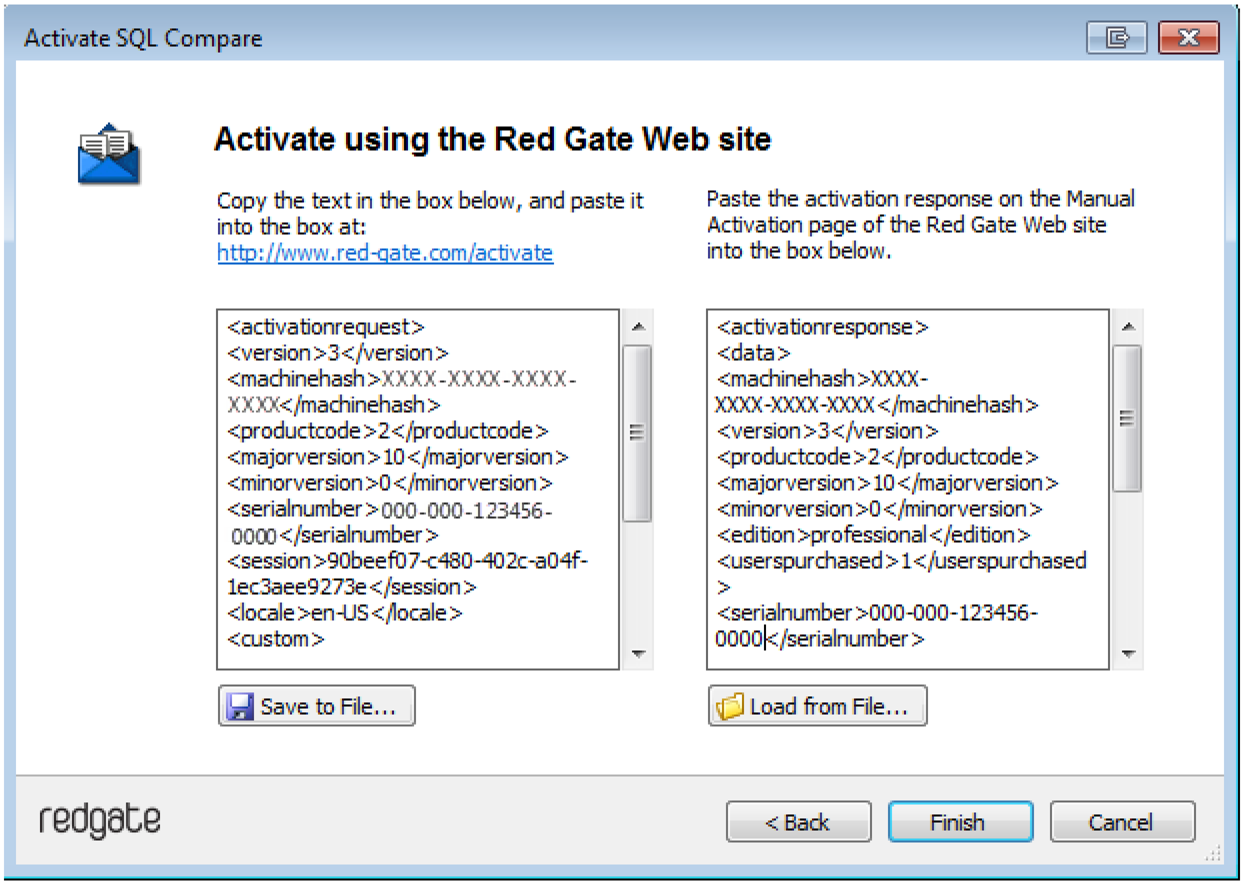
Paste the activation request into the box below. Make sure you paste all of the text.

```
<activationrequest>
<version>3</version>
<machinehash>XXXX-XXXX-XXXX-
XXXX</machinehash>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<serialnumber>000-000-123456-
0000</serialnumber>
<session>90beef07-c480-402c-a04f-
1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

Step 2

Copy the contents of this box into your product activation dialog box.

4. Click **Get Activation Response**.
5. When the activation response is displayed under **Step 2**, copy all of it.
   Alternatively you can save the activation response to a .txt file.
6. On the computer where the licensing and activation program is running, paste the activation response or if you saved it, load it from the file.



Activate using the Red Gate Web site

Copy the text in the box below, and paste it into the box at: http://www.red-gate.com/activate

Paste the activation response on the Manual Activation page of the Red Gate Web site into the box below.

```
<activationrequest>
<version>3</version>
<machinehash>XXXX-XXXX-XXXX-
XXXX</machinehash>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<serialnumber>000-000-123456-
0000</serialnumber>
<session>90beef07-c480-402c-a04f-
1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

```
<activationresponse>
<data>
<machinehash>XXXX-
XXXX-XXXX-XXXX</machinehash>
<version>3</version>
<productcode>2</productcode>
<majorversion>10</majorversion>
<minorversion>0</minorversion>
<edition>professional</edition>
<userspurchased>1</userspurchased
>
<serialnumber>000-000-123456-
0000</serialnumber>
```

7. Click **Finish**.
   The **Activation successful** page is displayed.
8. Click **Close**.
   You can now continue to use your product.

Deactivating

This page applies to several Redgate products, so the screenshots below may not match your product.

Download deactivation tool

You can use the deactivation tool to deactivate a serial number so you can reuse it on another computer. You can also use it to deactivate serial numbers for products you've uninstalled.

When you deactivate a serial number for a bundle of products, all the products in the bundle are deactivated. For information about what products are in your bundle, see Bundle history.

To deactivate a serial number, your computer must have an internet connection. If you can't deactivate a serial number, you can request additional activations for that serial number. You may need to do this if:

- your computer doesn't have an internet connection
- your network uses a proxy server that interrupts contact between the product and the Redgate activation server
- your serial numbers aren't displayed in the deactivation tool (eg if the product installation is corrupted)

Deactivating using the command line

Open a command prompt, navigate to the folder where your product executable file is located and run a command with the following syntax:

```
<productEXE> /deactivateSerial
```

For example:

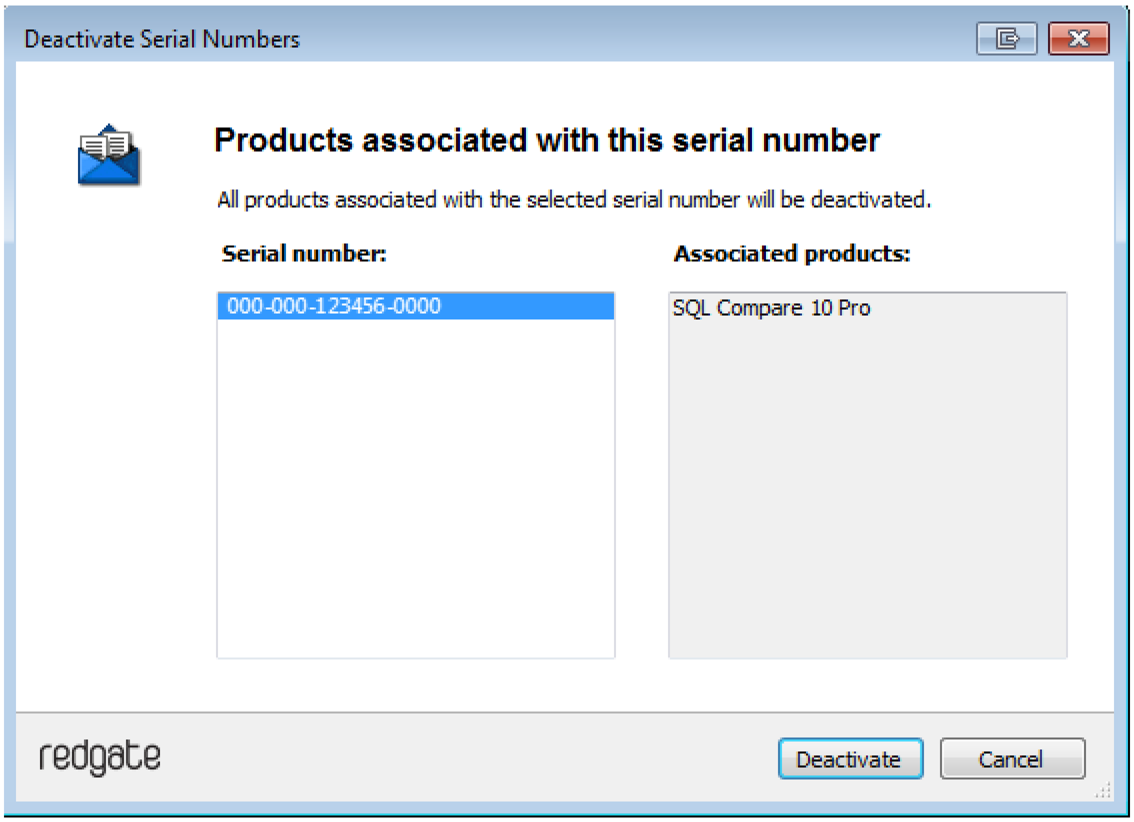
```
sqlcompare /deactivateSerial
```

The **Deactivate Serial Numbers** dialog box is displayed. Follow the instructions below.

Deactivating using the GUI

To deactivate your products:

1. Start the deactivation tool. To do this, either download the tool and run the executable file, or on the **Help** menu of the product, click **Deactivate Serial Number**.
   The **Deactivate Serial Numbers** dialog box is displayed. For example:


Deactivate Serial Numbers

Products associated with this serial number

All products associated with the selected serial number will be deactivated.

Serial number:

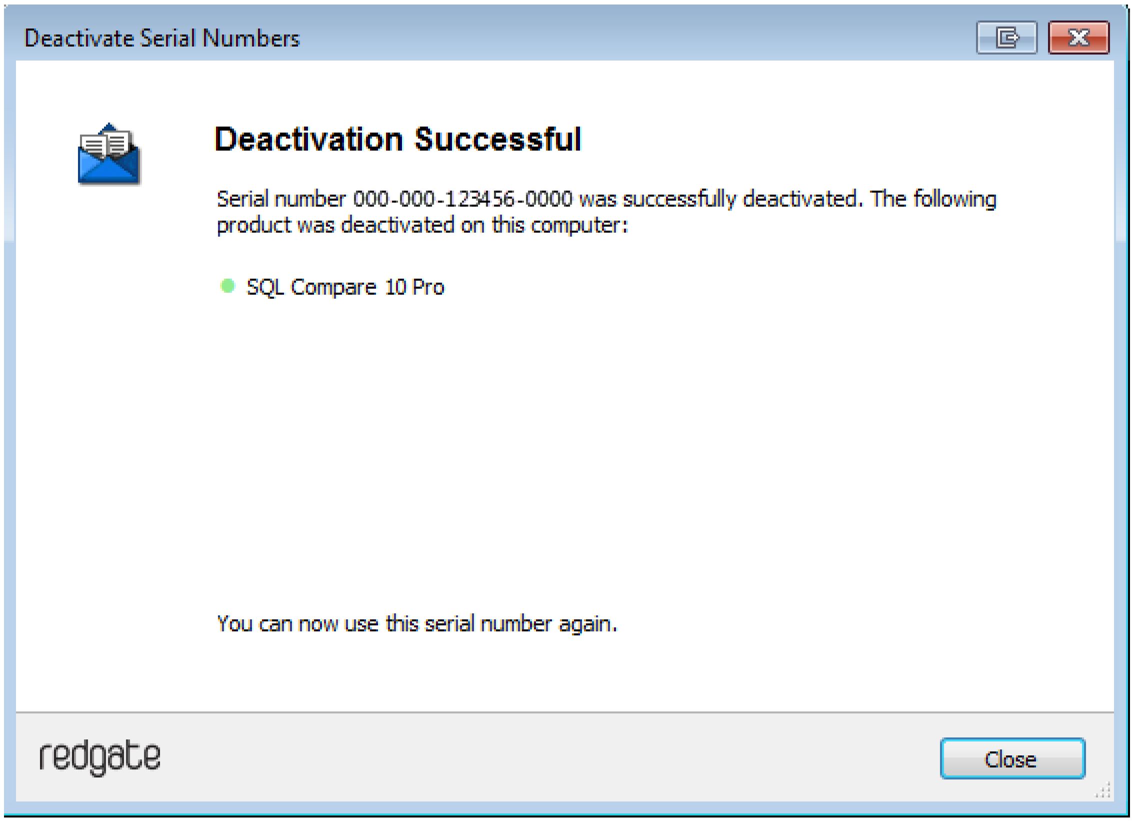
000-000-123456-0000

Associated products:

SQL Compare 10 Pro

If you're running the executable file, the dialog box displays all the serial numbers for Red Gate products that have been activated on your computer.
If the serial number is for a bundle, all the products in the bundle are displayed under **Associated products**.

2. Select the serial number you want to deactivate and click **Deactivate**.
   Your deactivation request is sent to the Red Gate activation server.
3. When your deactivation has been confirmed, the **Deactivation successful** page is displayed. For example:


Deactivate Serial Numbers

Deactivation Successful

Serial number 000-000-123456-0000 was successfully deactivated. The following product was deactivated on this computer:

● SQL Compare 10 Pro

You can now use this serial number again.

If there's a problem with your deactivation request, an error dialog box is displayed. For information about deactivation errors and how to resolve them, see Troubleshooting licensing and activation errors.

4. Click **Close**. You can now use this serial number on a different computer.

Troubleshooting licensing and activation

This page provides information about errors you may encounter when you activate Redgate products:

- The number of activations for this serial number has been exceeded
- This serial number has been disabled
- This serial number was for a trial extension
- This serial number is not registered with the activation server
- This serial number is not for <product name>
- This serial number is not for this version
- The activation request is in the wrong format
- The activation request contains an invalid machine hash
- The activation request contains an invalid session
- The activation request contains an invalid serial number
- The activation request contains an invalid product code or version number
- There's a problem deactivating your serial number
- This serial number is not activated on this computer
- Products not activated on this computer

The number of activations for this serial number has been exceeded

This error message is displayed when a serial number is activated on more computers than the number of licenses that were purchased for that serial number.

When you purchase products from Redgate, we send you an invoice that includes your serial numbers. The serial numbers enable you to activate the software a number of times, depending on how many licenses you purchased and the terms in the license agreement. When this limit is reached, you will see this error message.

To fix the problem, you can:

- deactivate the product on another computer to free up a license
- purchase more licenses
- request additional activations for your serial number

This serial number has been disabled

This error message is displayed when you try to activate a product using a serial number that Redgate has disabled.

When you upgrade a product, your existing serial numbers will be disabled and we will issue new ones with your invoice. If you cannot find your new serial numbers, you can review them at http://www.red-gate.com/myserialnumbers

Redgate will also disable serial numbers for non-payment of invoices or breach of the terms in the license agreement. If you think we have disabled your serial numbers in error, email licensing@red-gate.com

This serial number was for a trial extension

This error message is displayed when you have requested a trial extension and you try to reuse the serial number that was provided for the trial extension; trial extensions can be used one time only.

To continue using the product, you need to purchase it.

This serial number is not registered with the activation server

This error message is displayed when the serial number you entered does not exist on the Redgate activation server.

To find out your serial numbers, check your invoice or go to http://www.red-gate.com/myserialnumbers

This serial number is not for <product name>

This error message is displayed when the serial number you entered is not for the product you are trying to activate.

To find out your serial numbers, check your invoice or go to http://www.red-gate.com/myserialnumbers

This serial number is not for this version

This error message is displayed when the serial number you entered is for a different version of the product you are trying to activate.

If the serial number is for an older version of the product, and you don't have that version installed on your computer, you can download it from the Release notes and other versions page.

If you want to upgrade to the latest version of the product, go to the Upgrade center to get a quote or purchase an upgrade, or email sales@red-gate.com.

The activation request is in the wrong format

This error message is displayed:

- if your internet connection does not allow SOAP requests.
  Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed.
- if you are activating by email and there is a problem with the format of the activation request.
  Check that you copied and pasted all of the activation request.
  Alternatively, try using manual activation. Go to http://www.red-gate.com/activate and paste your activation request under **Step 1**.
- when you are using manual activation and there is a problem with the format of the activation request. If the format is incorrect, for example part of the request is missing, the Redgate activation server cannot process the request.
  Check that you copied and pasted all of the activation request.

For more information about activating manually, see Manual activation.

The activation request contains an invalid machine hash

This error message is displayed:

- if your internet connection does not allow SOAP requests.
  Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see Manual activation.
- when you are using manual activation and there is a problem with the format of the *machinehash* element in the activation request. The *machinehash* is a checksum of attributes from your computer. We use the *machinehash* to identify computers on which our products have been activated. If the format of the *machinehash* element is incorrect, the Redgate activation server cannot process the request.
  Check that you copied and pasted the activation request correctly.

The activation request contains an invalid session

This error message is displayed:

- if your internet connection does not allow SOAP requests.
  Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see Manual activation.
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the *session* element is incorrect, the Redgate activation server cannot process the request.
  Check that you copied and pasted the activation request correctly.

The activation request contains an invalid serial number

This error message is displayed:

- if your internet connection does not allow SOAP requests.
  Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see Manual activation.
- when you are using manual activation and there is a problem with the format of the activation request. If the format of the serial number is incorrect, the Redgate activation server cannot process the request.
  Check that you copied and pasted the activation request correctly.

The activation request contains an invalid product code or version number

This error message is displayed:

- if your internet connection does not allow SOAP requests.
  Try using manual activation; on the error dialog box, click **Activate Manually**, and then follow the instructions that are displayed. For more information, see Manual activation.
- when you are using manual activation and there is a problem with the format of the activation request. If the product code or version numbers are incorrect, the Redgate activation server cannot process the request.
  Check that you copied and pasted the activation request correctly.

There's a problem deactivating your serial number

This error message is displayed if your computer is not connected to the internet or your internet connection does not allow SOAP requests. You cannot deactivate a serial number if your computer does not have an internet connection.

Try deactivating again later. If the problem persists, contact your system administrator.

If you require more activations because you cannot deactivate your serial number, you can request them on the Request Extra Activations page.

This serial number is not activated on this computer

This error message is displayed when you try to deactivate a serial number that has not been activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the Request Extra Activations page.

Products not activated on this computer

This error message is displayed when you try to deactivate a serial number for a bundle of Redgate products and those products were not activated on your computer.

If you think the product installation on your computer is corrupt, you can try re-activating the product, and then deactivating the product again.

If you require more activations because you cannot deactivate your serial number, you can request them on the Request Extra Activations page.

Upgrading

Minor releases are free for all users. For example, if you have a license for version 7.0 of a product, you can upgrade to version 7.1 at no cost. W hen you download and install a minor release, the product is licensed with your existing serial number automatically.

Major releases are free for users with a current Support and Upgrades contract. For example, if you have a license for version 7 of a product, you can upgrade to version 8 at no cost. When you download and install a major release, the product is licensed with your existing serial number automatically.

If you don't have a current Support and Upgrades contract, installing a major release will start a free 14-day trial. You'll need to buy a new license and activate the product with your new serial number.

To check whether you have a current Support and Upgrades contract or see the cost of upgrading to the latest major version of a product:

- visit the Upgrade Center
- email sales@red-gate.com
- call:
  - 1 866 733 4283 (toll free USA and Canada)
  - 0800 169 7433 (UK freephone)
  - +44 (0)870 160 0037 (rest of world)

To check the latest version of a product, see Current versions.

How to upgrade

You can download the latest version of a product using Check for Updates, the Upgrade Center, or the Redgate website.

- If you download the latest version from the Upgrade Center or our website, you need to run the installer to upgrade the product.

  Some Redgate products are available as part of bundle. You can select which products you want to upgrade when you run the installer.

- If you use Check for Updates, the installer runs automatically.

You can install the latest *major* version of any product (other than SQL Backup Pro) on the same machine as the previous version. For example, you can run version 9 and version 10 in parallel. However, installing a *minor* release will upgrade the existing installation.

To revert to an earlier version, uninstall the later version, then download and install the version you want from the Release notes and other versions page. You can use a serial number for a later version to activate an earlier version.

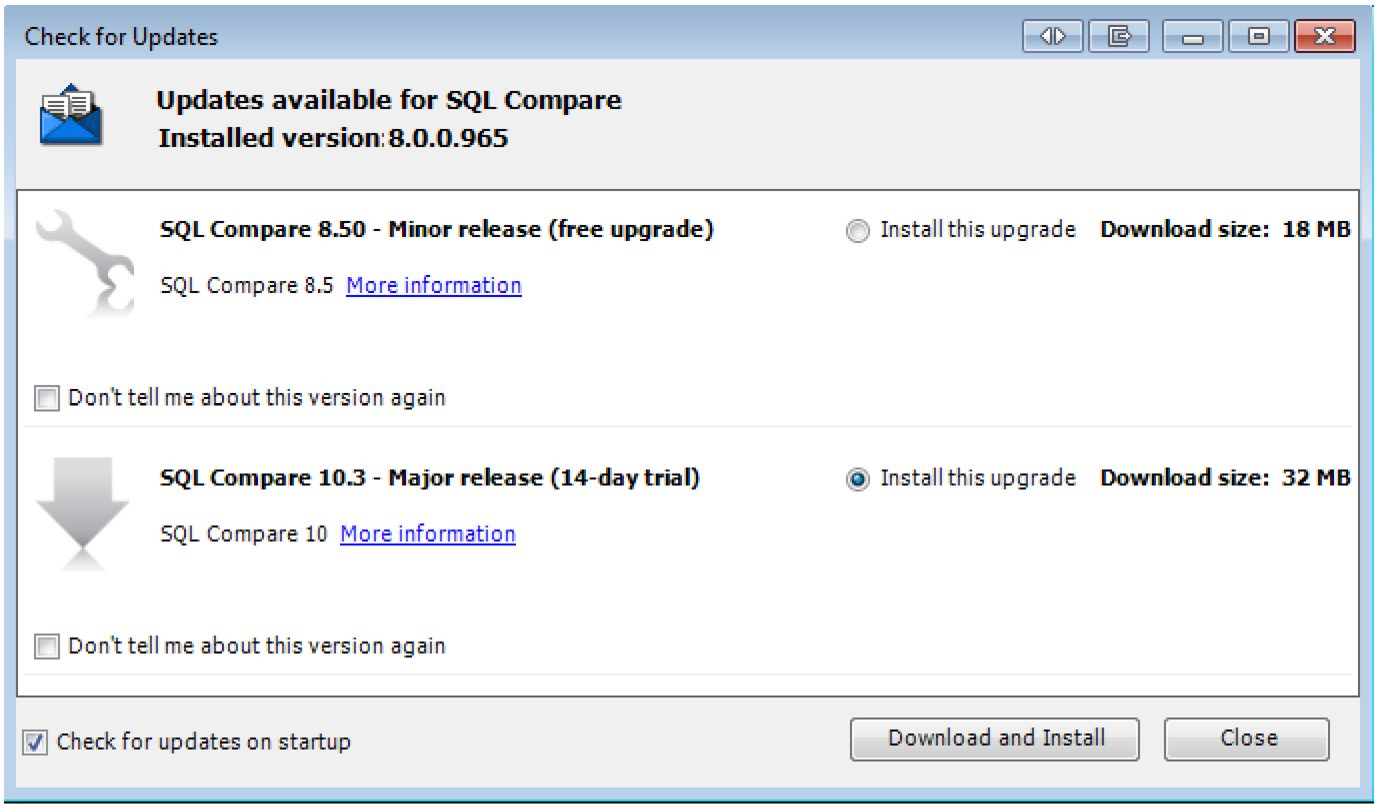
Using Check for Updates

This page applies to several Redgate products, so the screenshots below may not match your product.

The Check for Updates service checks whether a more recent version of the product is available to download. To use the service, your computer must have a connection to the internet. If your internet connection uses a proxy server, make sure your web browser connection settings are configured correctly.

The Check for Updates service doesn't work with automatic configuration scripts.

To check for updates for a Redgate product, on the **Help** menu, click **Check for Updates**. Any available updates are listed:



Check for Updates

Updates available for SQL Compare
Installed version: 8.0.0.965

SQL Compare 8.50 - Minor release (free upgrade) ( ) Install this upgrade **Download size: 18 MB**
SQL Compare 8.5 More information
[ ] Don't tell me about this version again

SQL Compare 10.3 - Major release (14-day trial) (●) Install this upgrade **Download size: 32 MB**
SQL Compare 10 More information
[ ] Don't tell me about this version again

[✓] Check for updates on startup

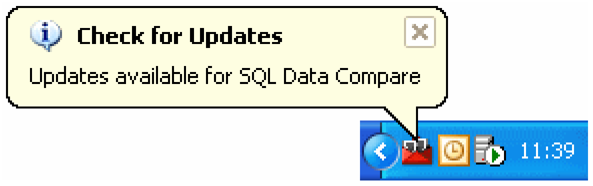
To view the full release details in your default web browser, click **More information**.

To get the update, click **Download and Install**. If you have a choice of updates, choose by selecting **Install this upgrade**, and then click **Download and Install**.

The installer will ask you to close the program. If you're upgrading an add-in, you'll also be asked to close the host program (SQL Server Management Studio, Visual Studio or Query Analyzer).

About the Check for Updates service

When you start the application, the Check for Updates service informs you automatically when there are updates available:



Check for Updates
Updates available for SQL Data Compare

If you don't want to receive these notifications for the product, clear the **Check for updates on startup** check box.

If you don't want the Check for Updates service to inform you about a particular update again, select the **Don't tell me about this version again** check box. The Check for Updates service will still inform you of new updates when they become available.

Troubleshooting Check for Updates errors

For details about how to use the Check for Updates service, see Using Check for Updates.

Error: There is a problem saving the download file to your computer

This error message is displayed if:

You don't have enough disk space

The Check for Updates service downloads the updates to the location defined by the *RGTEMP* environment variable, or the *TMP* variable if the *RGTEMP* variable doesn't exist.
If you don't have enough disk space, you can change the environment variable to a location with more space.

> Changing the *RGTEMP* or the *TMP* variables will affect other programs that use those variables. The *RGTEMP* variable affects only Redgate programs. For information about environment variables, see your Windows documentation.

There's a problem with permissions on your computer

The Check for Updates service downloads the updates to the location defined by the *RGTEMP* environment variable, or the *TMP* variable if the *RGTEMP* variable does not exist. If your user account doesn't have permissions to write to the location specified by these environment variables, contact your system administrator.

There's a problem with the download file on the Redgate web server

Contact Redgate support.

Error: There is a problem with the network connection

This error message is displayed if:

Your internet connection dropped while the Check for Updates service was downloading the updates

Try checking for updates again later.

Proxy authentication failed

Check your user name and password.

Your computer can't connect to the Check for Updates service.

Contact your system administrator. If you're using a proxy server, check it's configured correctly (see Control Panel > Internet Options > Connections).

> The Check for Updates service doesn't work with automatic configuration scripts.

There's a problem with the download file on the Redgate web server

Contact Redgate support.

Creating the package

Whenever you create a database package, you set up a *project*. A project contains:

- the information required to connect to the database you want to package
- information about which objects and data you want to include in the package
- the package details such as name, location, type, and compression
- optional deployment notes
- optional details about the database that will be created, such as the default name and size

You can create a new project each time you create a package. Alternatively, if you will be using the same settings repeatedly, you can save the current project to a file. You can then open it at a later date to make it the current project. You can also edit the current project if required.

You are recommended to run the project using the default packaging options and review the results before you change any options. If you want to change your packaging options before you run the next project, if necessary click **Cancel** on the SQL Packager Wizard, and set the options befor e you proceed.

Packaging options are not saved as part of the project.

To create a new project:

1. If the SQL Packager Wizard is not already displayed, click
   New Project. The Packager Wizard opens.If you have unsaved changes in the current project, you are prompted to save it.
2. In the SQL Packager Wizard, choose a project type.
3. If you are not packaging an existing SQL script, specify the package contents.
4. Review any generated SQL scripts for the object and data creation.
5. Create an .EXE (.NET executable) or a C# project. SQL Packager packages the database. Alternatively, you can launch the package creation script in your default SQL editor, or save the script.
   You can then save the project if required.

Working with projects

You can create a new project each time you create a package. Alternatively, if you will be using the same settings repeatedly, you can save the current project to a file. You can then open it at a later date to make it the current project. You can also edit the current project if required.

Saving the current project

When you have packaged a database, you can save a new project so that you can re-use the settings at a later date, or you can save an existing project to a new name. Projects are saved with the extension .sqlpkg

- On the **File** menu, click **Save Project As**.
  A standard Windows **Save As** dialog box is displayed.

If you have edited the current project, an asterisk (*) is shown in the title bar; to save the edited project to the same name:

- On the **File** menu, click **Save Project**; alternatively, click

Editing the current project

You can edit the current project and package the database again.

1. Click
   Edit Project.
2. In the Packager Wizard, change the package specification as required.
3. Review the generated SQL scripts for the object and data creation.
4. Generate an .EXE or a C# project.SQL Packager packages the database. An asterisk (*) is displayed in the title bar to indicate that the project has been edited.

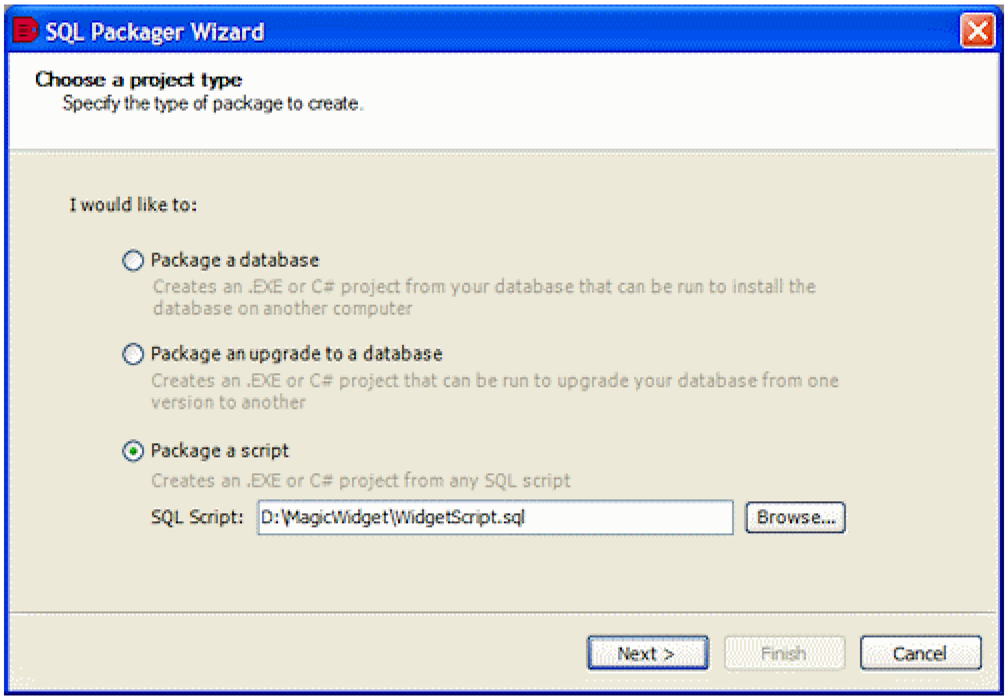
Opening an existing project

1. If the Packager Wizard is displayed, click **Cancel** to close it.
2. Click
   Open Project.
   If you have unsaved changes in the current project, you are prompted to save it.
3. In the **Open** dialog box, choose the project and click **Open**.
   The Packager Wizard is displayed with the project's settings.

To open a recently-used project, on the **File** menu, click the name of the project.

Choosing a project type

SQL Packager allows you to package a database, a database upgrade, or any SQL script as an .EXE or a C# project.


SQL Packager Wizard
Choose a project type
Specify the type of package to create.
I would like to:
Package a database
Creates an .EXE or C# project from your database that can be run to install the database on another computer
Package an upgrade to a database
Creates an .EXE or C# project that can be run to upgrade your database from one version to another
Package a script
Creates an .EXE or C# project from any SQL script
Browse...
Next >
Finish
Cancel

( ) Package a database
( ) Package an upgrade to a database
[✓ Package a script]

SQL Script: D:\MagicWidget\WidgetScript.sql

Package a database

For information on packaging a database, see Worked example: packaging a database as an .EXE.

Package an upgrade to a database

For information on packaging an upgrade to a database, see Worked example: packaging an upgrade as a C# project.

Package a script

Note that when packaging a script, SQL Packager does not check the SQL syntax of the script. If the script contains any errors, you will encounter those errors when you run the package.

1. On the **Choose a project type** page of the Packager Wizard, select **Package a script**.
2. Click **Browse** to select the script you want to package, and click **Next**.

The Specify package contents page of the Packager Wizard is displayed.

See one of the worked examples for information on using the Packager Wizard.

Specifying the package contents

Whenever you package a database, SQL Packager requires information to connect to the database you want to package, and information about the database objects and data you want to include in that package. You enter this information using the SQL Packager Wizard.

The information you enter in the SQL Packager Wizard is saved in the current project. For more information about projects, see Working with projects.

Choosing the databases

1. If you have chosen to **Package a database** on the Choose a project type page, the **Choose database to package** page is displayed:

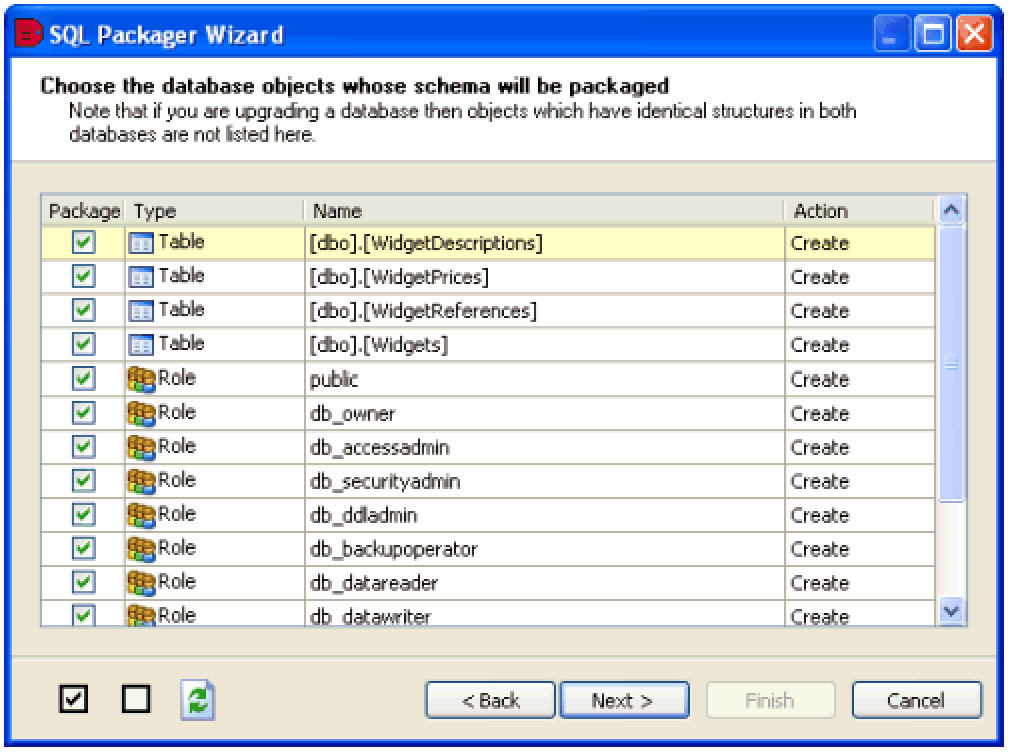
   Alternatively, if you have chosen **Package an upgrade to a database**, the **Choose databases to package into an upgrade** page is displayed:

2. In the relevant **Server** box, type or select the name of the SQL Server.
   If you experience problems selecting a SQL Server that is not running on the LAN, for example if you are accessing the SQL Server via an internet connection, you may need to create an alias to the SQL Server using TCP/IP (refer to your SQL Server documentation for details). You can then type the alias name in the **Server** box to connect to the remote SQL Server.
   To refresh the **Server** list, right-click the box and click **Refresh**, or scroll to the top of the list and click **Refresh**.
3. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
4. In the **Database** box, type or select the database that you want to package.
   To refresh the **Database** list, right-click in the box and click **Refresh**, or scroll to the top of the list and click **Refresh**.
5. If you are packaging an upgrade, in **Database to upgrade** enter the SQL Server and database details.
6. Click **Next**. SQL Packager displays a message dialog box while it analyzes the database structure.
   If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that you choose the databases.
7. If necessary, click **OK** to close the message box.

Choosing the database objects to package

If you are creating a package for a new database, SQL Packager lists the objects in the database that you can select for packaging.

If you are creating a package for upgrading an existing database, SQL Packager compares the previous version database with the latest version database to identify the changes to be made to the database structure to make the databases identical. SQL Packager lists the objects whose structure differs.



Choose the database objects whose schema will be packaged
Note that if you are upgrading a database then objects which have identical structures in both databases are not listed here.

Package	Type	Name	Action
✓	Table	[dbo].[WidgetDescriptions]	Create
✓	Table	[dbo].[WidgetPrices]	Create
✓	Table	[dbo].[WidgetReferences]	Create
✓	Table	[dbo].[Widgets]	Create
✓	Role	public	Create
✓	Role	db_owner	Create
✓	Role	db_accessadmin	Create
✓	Role	db_securityadmin	Create
✓	Role	db_ddladmin	Create
✓	Role	db_backupoperator	Create
✓	Role	db_datareader	Create
✓	Role	db_datawriter	Create

You select the objects to package by selecting or clearing the appropriate check boxes in the **Package** column. By default, the first time you run a project all objects are selected for packaging. To select all objects, click

☑

; to clear all of the check boxes, click

☐

You may see the following types of object:

	Tables			User Defined Types
	Rules			Users
	Views			Functions
	Defaults			Roles
	Stored Procedures			Full Text Catalogs

For SQL Server 2008 or SQL Server 2005, the following object types may also be shown:

	Assemblies			Queues
	Asymmetric Keys			Routes
	Certificates			Schemas
	Contracts			Services
	DDL Triggers			Service Bindings
	Event Notifications			Symmetric Keys

	Message Types			Synonyms
	Partition Functions			XML Schema Collections
	Partition Schemes			Full Text Stoplist (SQL Server 2008 only)

SQL Server 2008 and SQL Server 2005 severely restrict access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Packager can package only the permissions of certificates and asymmetric keys; symmetric keys cannot be packaged. To ignore all certificates, symmetric keys, and asymmetric keys, select the Ignore certificates, symmetric and asymmetric keys schema packaging option.

If you are upgrading a database, note that:

- Objects that are the same but have different owners are treated as different objects. For example, if a stored procedure exists in both databases and is identical except for its owner, it is considered to be a completely different object.
- For SQL Server 2000, differences in database-level permissions are not detected by SQL Packager. For example, if you have used SQL Server Enterprise Manager to set up permissions for your database, such as GRANT CONNECT or GRANT BACKUP, those permissions are not considered; however, permissions on objects are detected. If you want to include database-level permissions in your database package, you are recommended to use roles. For SQL Server 2008 and SQL Server 2005 differences in database-level permissions are detected.
- If the database contains an encrypted user-defined function, stored procedure, trigger, or view and you have system administrator permissions, SQL Packager decrypts the object and you can view its internal SQL in the schema packaging script. If you do not have system administrator privileges, the encrypted object cannot be displayed or upgraded.SQL Packager cannot decrypt views, stored procedures, functions, DML triggers, and DDL triggers that are encrypted in a SQL Server 2008 or SQL Server 2005 database. Therefore, SQL Packager cannot compare the objects; if an encrypted object exists in both databases, SQL Packager assumes that they are different, but will not be able to upgrade them.
- Stored procedures that are for replication are not compared or displayed.
- SQL Packager does not compare extended stored procedures.

For each object, the **Action** column indicates the action that will be taken on the object. If you are packaging a database to create a new one, the action will be **Create** for all objects. However, for an upgrade, note that the package does not only add new objects to the upgraded database; it may also **Alter** or **Drop** objects to make the databases identical.

If you are editing an existing project and the structure of the database has changed since you previously packaged it, you may need to click

to update the page with the new structure.

When you have selected the objects that you want to package, click **Next**.

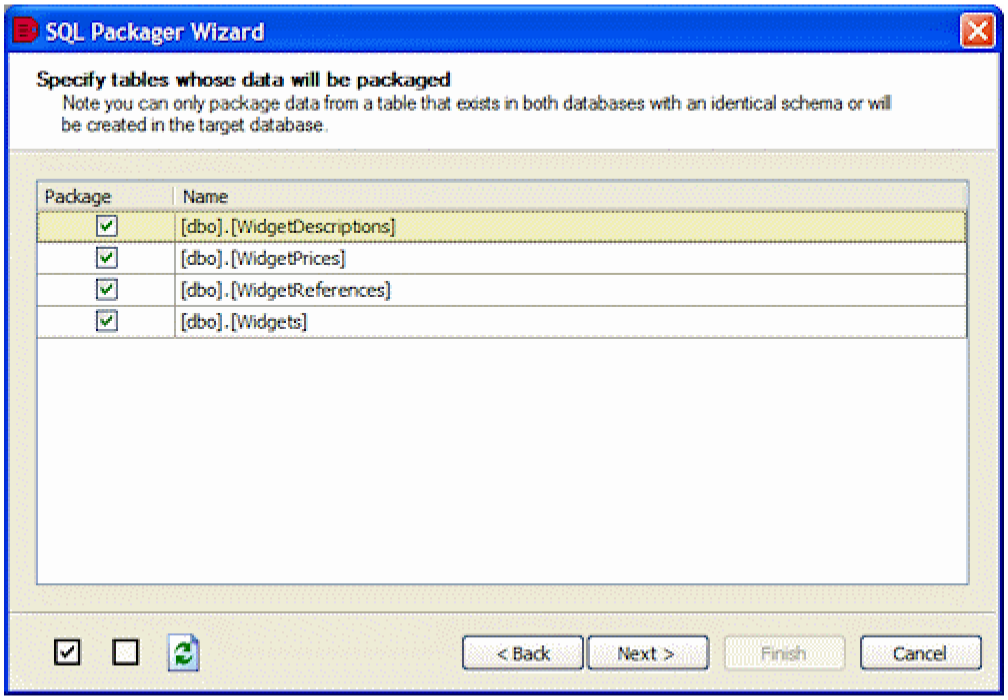
Choosing the data to package

SQL Packager lists the tables in the database for you to select for data packaging.

If you are creating a package for upgrading an existing database, you can select tables for data packaging only if they have:

- similar names
  You can set SQL Packager so that it ignores the case of object names, and spaces or underscores in object names by using the data packaging options.
- the same owner (case-sensitive)
- similar structures
- a primary key, unique index, or unique constraint that matches in both databases
  SQL Packager uses the key, index, or constraint to determine which records correspond with each other. If more than one matching primary key, unique index, or unique constraint exists for a table, SQL Packager selects the most applicable (for example, a primary key is used in preference to a unique index).

You can also select a table for data packaging if it does not exist in the database you are upgrading, and you have chosen to package the table's structure in the previous page of the wizard.



Specify tables whose data will be packaged
Note you can only package data from a table that exists in both databases with an identical schema or will be created in the target database.

Package	Name
✓	[dbo].[WidgetDescriptions]
✓	[dbo].[WidgetPrices]
✓	[dbo].[WidgetReferences]
✓	[dbo].[Widgets]

You select the tables for data packaging by selecting or clearing the appropriate check boxes in the **Package** column. If you are creating a package for a new database, the package will insert data into the tables you have selected. However, for an upgrade, the package not only inserts data into tables, it may also update or delete rows to make the databases identical.

By default, the first time you run a project, all available tables are selected. To select all objects, click
☑
; to clear all of the check boxes, click
☐

If you are editing an existing project and the databases' data has changed since you previously packaged it, you may need to click
🔄
to update the page.

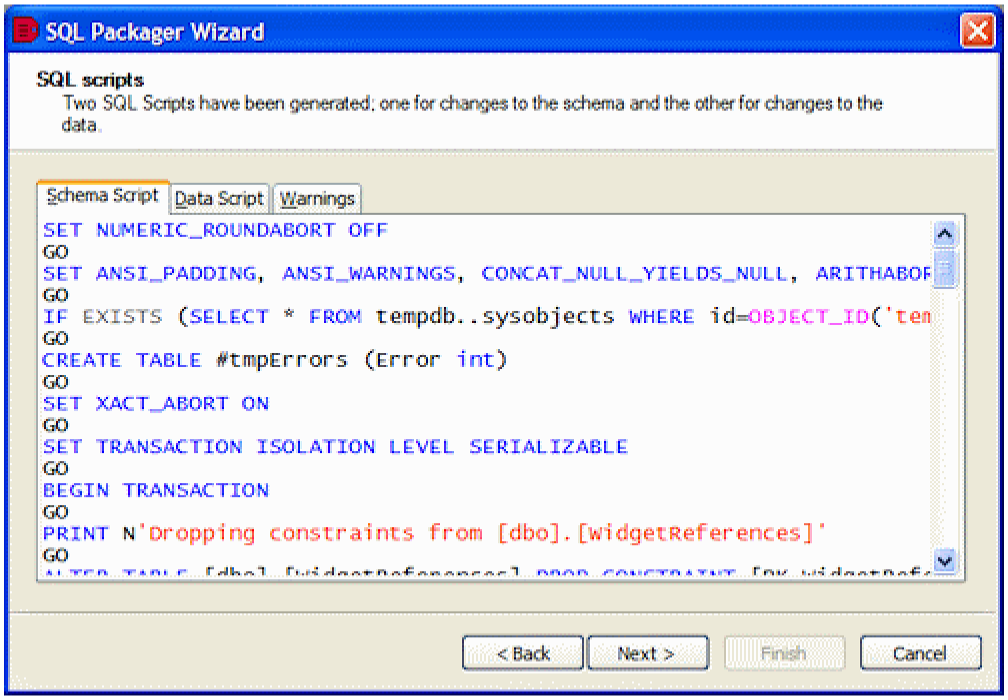
When you have selected the tables for data packaging, click **Next**. SQL Packager displays a message dialog box while it generates the SQL script.

If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that it generates the SQL script.

If necessary, click **OK** to close the message box. The **SQL scripts** page of the Packager Wizard is displayed; for details, see Previewing the SQL scripts.

Previewing the SQL scripts

The SQL Packager Wizard displays the SQL scripts for creating or modifying the database structure and data.



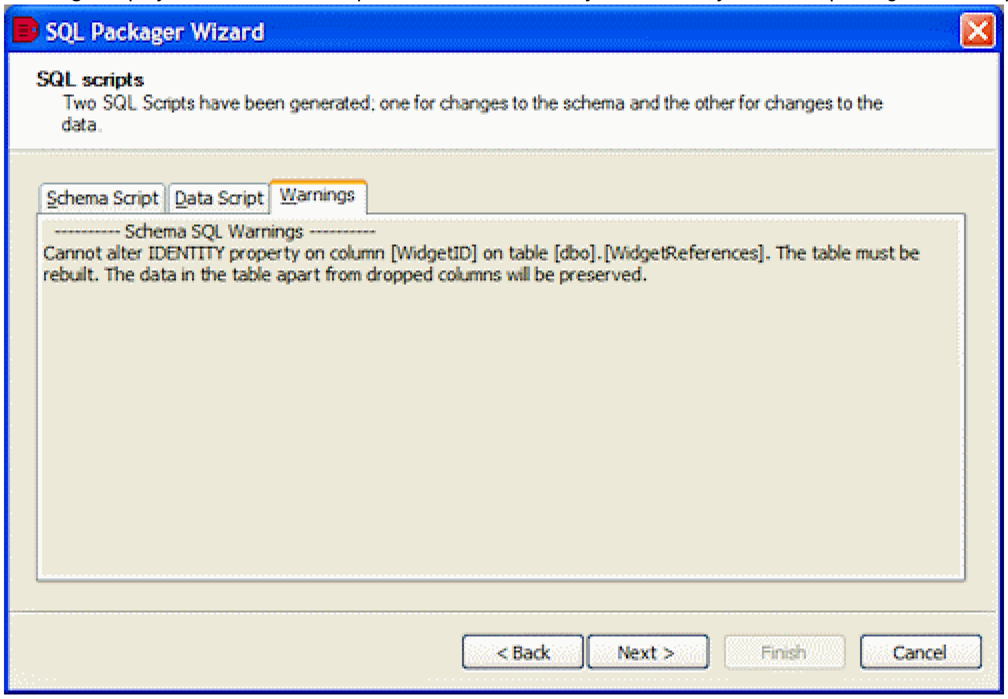
SQL scripts

Two SQL Scripts have been generated; one for changes to the schema and the other for changes to the data.

```
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABOR
GO
IF EXISTS (SELECT * FROM tempdb..sysobjects WHERE id=OBJECT_ID('tem
GO
CREATE TABLE #tmpErrors (Error int)
GO
SET XACT_ABORT ON
GO
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
GO
BEGIN TRANSACTION
GO
PRINT N'Dropping constraints from [dbo].[WidgetReferences]'
GO
```

The **SQL scripts** page displays the following tabs:

- **Schema Script** shows the SQL script to create or modify the database structure.
- **Data Script** shows the SQL script to create or modify the data.
- **Warnings** displays details about unexpected behavior that may occur when you run the package, for example:



SQL scripts

Two SQL Scripts have been generated; one for changes to the schema and the other for changes to the data.

```
---------- Schema SQL Warnings ----------
Cannot alter IDENTITY property on column [WidgetID] on table [dbo].[WidgetReferences]. The table must be
rebuilt. The data in the table apart from dropped columns will be preserved.
```

You can:

- Search a SQL script:
  Click the **Schema Script** or **Data Script** tab, right-click the SQL code, and click **Find**; a standard Windows **Find** dialog box is displayed.
- Copy the SQL scripts, summary details, or warning information for use in another application:
  Select the required text, right-click, and then click **Copy**.
  Alternatively, right-click the text, click **Select All**, then right-click, and click **Copy**.

- Save the scripts or launch them in a SQL editor on the next page of the SQL Packager Wizard.

When you have reviewed the SQL scripts and any warnings, click **Next** to specify the package type, or choose to save or launch the scripts:

- To create an .EXE (.NET executable), see Generating an .EXE.
- To create a C# project, see Generating a C# project.If you will want to customize the package, you are recommended to create a C# project.

Generating a package

When you have finished reviewing the SQL scripts, specify the package type:

- package as an .EXE (as a .NET executable)
- package as a C# project

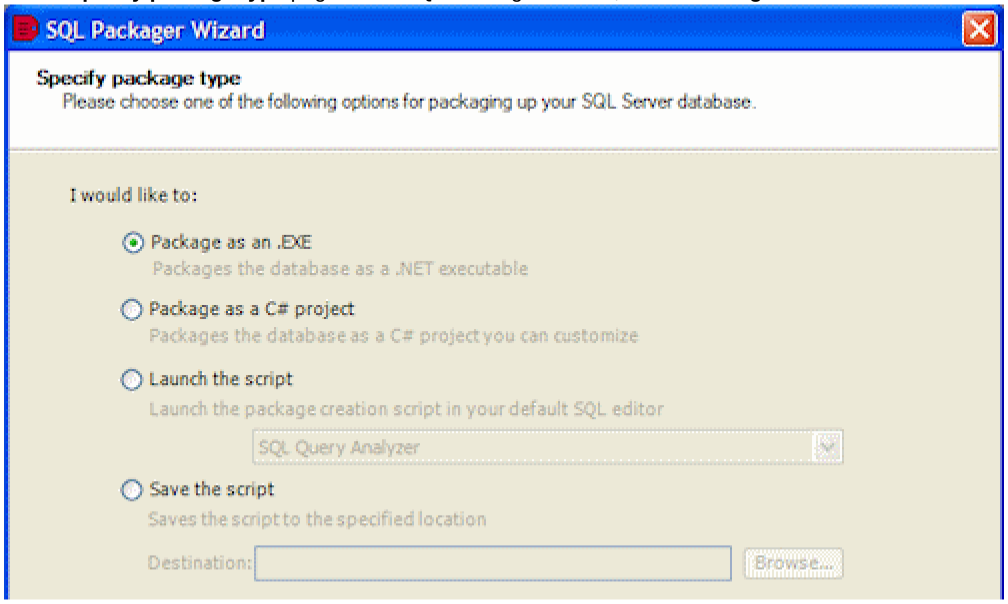
From the Specify package type page of the SQL Packager Wizard, you can also launch the script in your default SQL editor, or save the script.

Generating a .NET executable

When you have chosen the database, objects, and data, that you want to package and previewed the SQL scripts, you can define the parameters for the .EXE.

To create an .EXE:

1. On the **Specify package type** page of the SQL Packager Wizard, select **Package as an .EXE**.



SQL Packager Wizard

Specify package type
Please choose one of the following options for packaging up your SQL Server database.

I would like to:

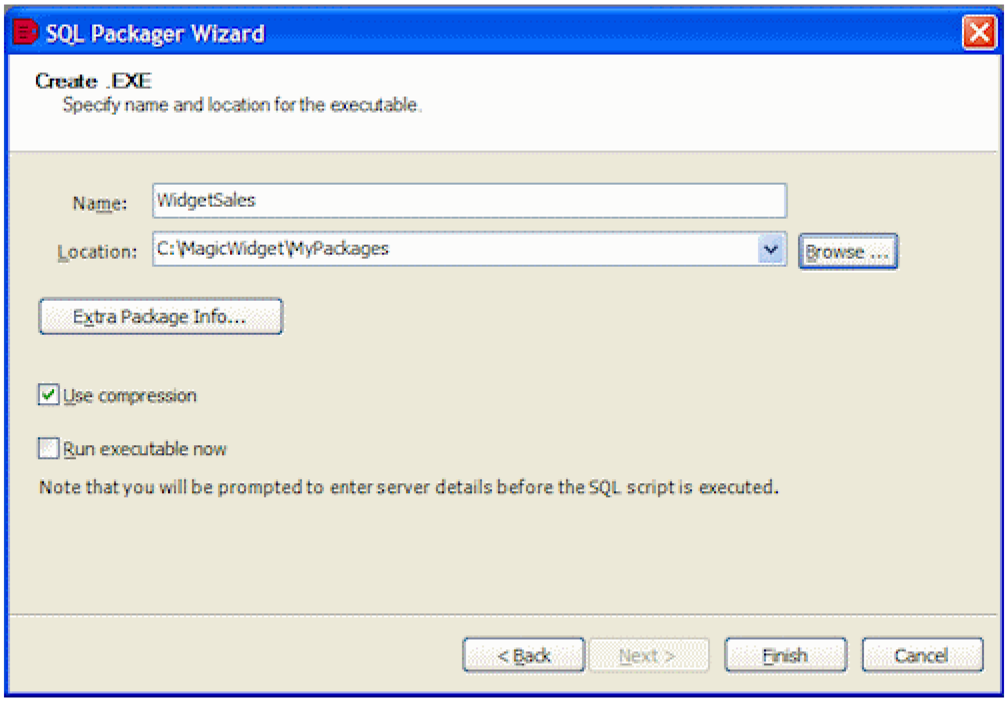
(●) Package as an .EXE
Packages the database as a .NET executable

( ) Package as a C# project
Packages the database as a C# project you can customize

( ) Launch the script
Launch the package creation script in your default SQL editor
SQL Query Analyzer

( ) Save the script
Saves the script to the specified location
Destination: Browse...

2. Click **Next**.



SQL Packager Wizard

Create .EXE
Specify name and location for the executable.

Name: WidgetSales
Location: C:\MagicWidget\MyPackages

[✓] Use compression
[ ] Run executable now

Note that you will be prompted to enter server details before the SQL script is executed.

3. On the **Create .EXE** page, in the **Name** box, type a name for the package.
   If you create a package with a file name that already exists, SQL Packager automatically assigns a different file name for the package. For example, if *SQLPackage* exists, the default name for the new package is *SQLPackage1*. You can turn off this feature by using the packaging options.
4. In the **Location** box, type or select the path for the package, or click **Browse** to choose the folder or create a new folder.
   You can change the default location by using the packaging options.
   If you want to add information to be seen when the package is run, or if you are creating a package for a new database and you want to specify the database properties, click Extra Package Info and enter the details as required.
5. To compress the generated files, select the **Use compression** check box.
   The package is usually compressed to approximately 75% of its original size.

6. To run the executable immediately, select the **Run executable now** check box; to create the executable without running it, clear the **Run executable now** check box.
7. Click **Finish**.
   A message dialog box is displayed to confirm that the executable has been created at the location you specified.
8. Click **OK** to close the message dialog box.If you chose to run it immediately, SQL Packager launches the **Run Package** dialog box. For more information, see Running the package.

For large databases, additional dynamic-link library (.dll) files are also created.

Generating a C# project

When you have chosen the database, objects, and data that you want to package and previewed the SQL scripts, you can define the parameters for the C# project.

You should create a C# project if you want to customize the package. For example, you can edit the forms created in the project to customize the appearance of the graphical user interface that is displayed when you run the package.

To create a C# project:

1. On the **Specify package type** page of the SQL Packager Wizard, select **Package as a C# project**.
2. Click **Next**.
3. On the **Create C# Project** page, in the **Name** box, type a name for the project.
   If you create a package with a file name that already exists, SQL Packager automatically assigns a different file name for the package. For example, if *SQLPackage* exists, the default name for the new package is *SQLPackage1*. You can turn off this feature by using the packaging options.
4. In the **Location** box, type or select the path for the project, or click **Browse** to choose the folder or create a new folder.
   You can change the default location by using the packaging options.
   If you want to add information to be seen when the package is run, or if you are creating a package for a new database and you want to specify the database properties, click Extra package info and enter the details as required.
5. To compress the package, select **Compress generated SQL resource files**.
   The package is usually compressed to approximately 75% of its original size.

   > If you compress the package, you will not be able to add resource files to the project or edit the existing resource files.

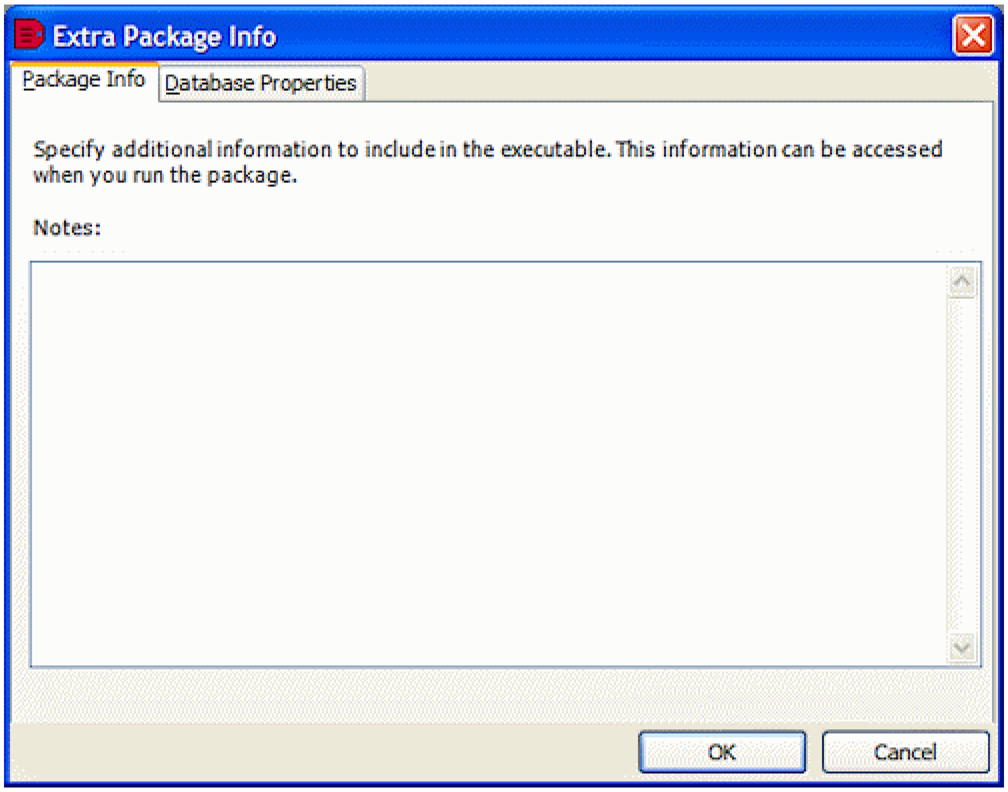
6. To open the project in Microsoft Visual Studio immediately, select the **Open project in Visual Studio** check box; to create the project without opening it, clear the check box.
7. Click Finish.
   A message dialog box is displayed to confirm that the project has been created at the location you specified.
8. Click OK to close the message dialog box.

If you chose to open it immediately, SQL Packager launches the project in Visual Studio. For more information, see Running the package.

Entering extra package information

When you create an .EXE or a C# project, you can specify extra information about the package, which can be viewed when the package is run.

To display the **Extra Package Info** dialog box, on the Create .EXE or Create C# Project page of the Packager Wizard, click **Extra Package Info**.

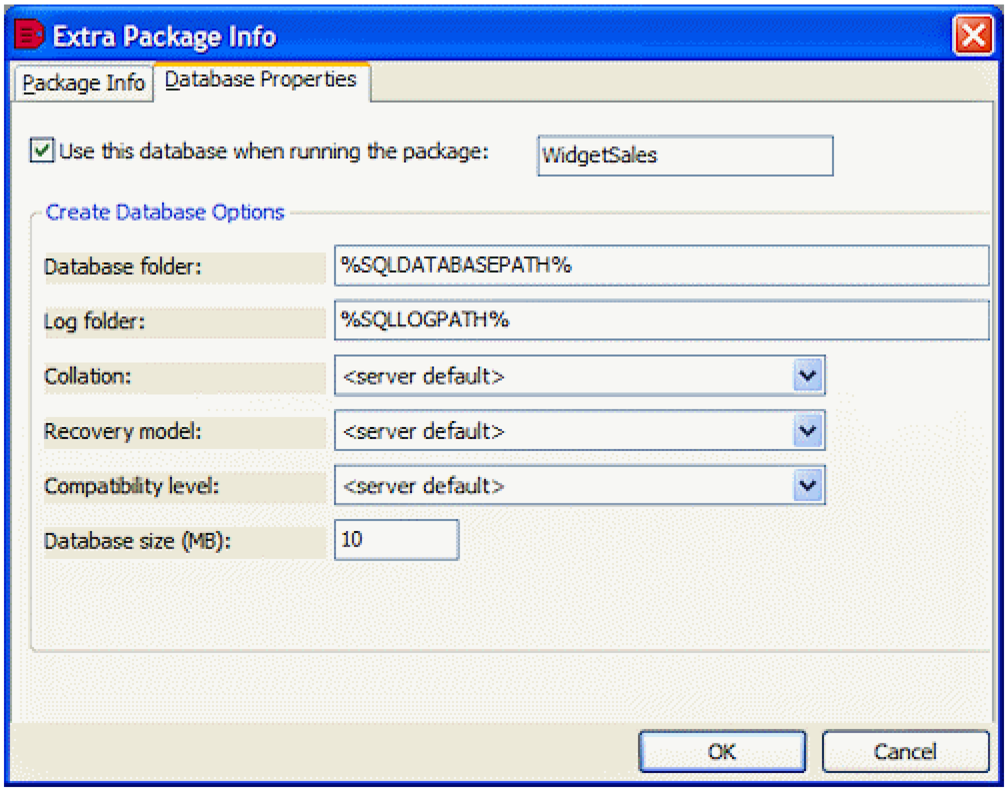


Specify additional information to include in the executable. This information can be accessed when you run the package.

Notes:

You can type free form notes on the **Package Info** tab.

If you are creating a package for a new database, you can specify the database properties by clicking the **Database Properties** tab.



[✓] Use this database when running the package: WidgetSales

Database folder:	%SQLDATABASEPATH%
Log folder:	%SQLLOGPATH%
Collation:	<server default>
Recovery model:	<server default>
Compatibility level:	<server default>
Database size (MB):	10

Enter the details as required. Note the following:

- **Collation** is dependent on the target system.
- **Compatibility level** must be compatible with the latest version database that you specified when you chose the database to package.

You can access these details at runtime from the **Run Package** dialog box, and amend them if required. For more information, see Running the Package.

For upgrades, you can specify only the database name to be used when the package is run.

Running the package

When SQL Packager has generated the package, you can run it using a graphical user interface or from the command line.

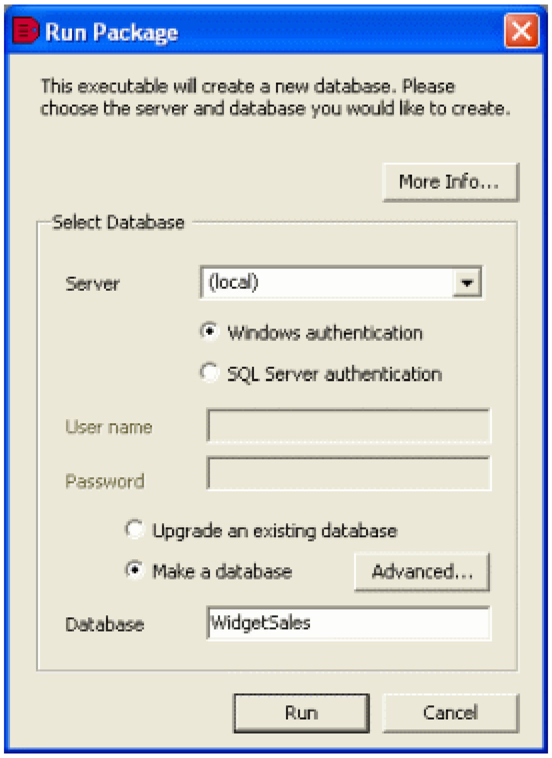
> The graphical user interface may differ from the interface described in this topic if it has been customized.

Using the graphical user interface to create a database

To create a new database:

1. Display the **Run Package** dialog box:
   - For an .EXE package, run the executable in the usual way.
     For example, double-click the .exe file in Microsoft Windows Explorer.
   - For a C# project, open the project in Visual Studio, press **F5** to run it, or on the **Debug** menu, click **Start.**
     Alternatively, compile the project and run the executable in the usual way.

   The **Run Package** dialog box is displayed.

   Run Package

   This executable will create a new database. Please choose the server and database you would like to create.

2. To view any notes that were added to the package when it was generated, or to see a summary of the SQL scripts and any warnings, click **More Info**.
   The **More Information** dialog box is displayed with the information. Click **OK** to close it.
3. In the **Server** box, type or select the name of the SQL Server on which you want to create the database.
   If you experience problems selecting a SQL Server that is not running on the LAN, for example if you are accessing the SQL Server via an internet connection, you may need to create an alias to the SQL Server using TCP/IP (refer to your SQL Server documentation for details). You can then type the alias name in the **Server** box to connect to the remote SQL Server.
4. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
5. Do one of the following:
   - To create a new database, click **Make a database** (the default) and type a name for the database in the **Database** box.
     If you want to change the default database properties, click **Advanced** and enter the details as required. For more information about database properties, see Entering extra package information.

     > If you want your database to use filegroups or full-text processing you should create the database manually, and use the **Upgrade an existing database** option.

   - To populate an empty database that you have already created, click **Upgrade an existing database** and in the **Database** box, type or select the name of the database.
6. Click **Run**.
   A message dialog box is displayed for you to confirm that you want to continue. A second message dialog box confirms that the package

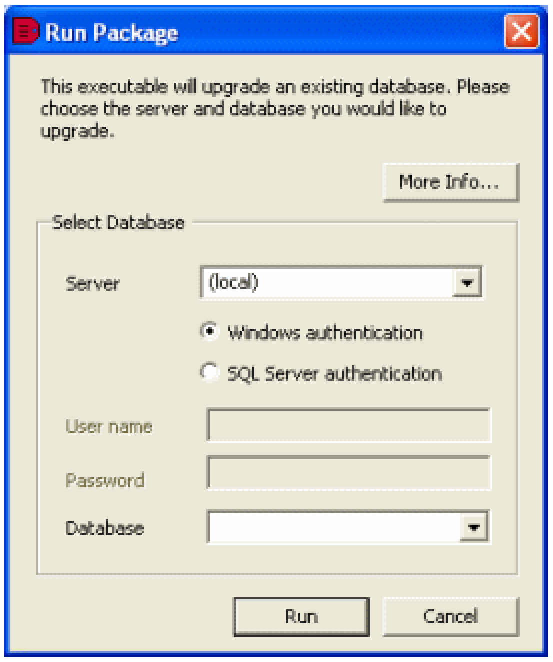
has run successfully.

Using the graphical user interface to upgrade a database

To upgrade an existing database:

1. Display the **Run Package** dialog box:
   - For an .EXE package, run the executable in the usual way
     For example, double-click the .exe file in Windows Explorer.
   - For a C# project, open the project in Visual Studio .NET, press **F5** to run it, or on the **Debug** menu, click **Start.**
     Alternatively, compile the project and run the executable in the usual way.

   The **Run Package** dialog box is displayed.

   
   This executable will upgrade an existing database. Please choose the server and database you would like to upgrade.

   Server: (local)
   (● Windows authentication) ○ SQL Server authentication

2. To view any notes that were added to the package when it was generated, or to see a summary of the SQL scripts and any warnings, click **More Info**.
   The **More Information** dialog box is displayed with the information. Click **OK** to close it.
3. In the **Server** box, type or select the name of the SQL Server for the database you are upgrading.
   If you experience problems selecting a SQL Server that is not running on the LAN, for example if you are accessing the SQL Server via an internet connection, you may need to create an alias to the SQL Server using TCP/IP (refer to your SQL Server documentation for details). You can then type the alias name in the **Server** box to connect to the remote SQL Server.
4. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
5. In the **Database** box, type or select the name of the database that you want to upgrade.
6. Click **Run**.
   A message dialog box is displayed for you to confirm that you want to continue. A second message dialog box confirms that the package has run successfully.

Using the command line

When you run the package from the command line, the following options are available:

`/server:<server>`

The name of the SQL Server. The default is (local).

`/server:<server>\<instance>`

The name of the SQL Server and instance. The default is (local).

`/database:<database>`

The name of the database that you want to create or upgrade.

`/username:<username>`

The user name for the database for SQL Server authentication.

`/password:<password>`

The password for the database for SQL Server authentication.

`/quiet`

Runs the package without displaying the graphical user interface.

`/makedatabase`

Creates the database on the specified SQL Server.

`/makeupgrade`

Upgrades the database on the specified SQL Server.

`/presql:<filename>`

Specifies a SQL script to run before the package is run.

`/postsql:<filename>`

Specifies a SQL script to run after the package has been run. Note that the script is always run, even if errors occur when the package is run.

For example, to create a database called **MyDatabase** on SQL Server **MyServer** using SQL Server authentication, navigate to the folder that contains the package, and at the command prompt, type:

```
MyPackage.exe /server:MyServer /database:MyDatabase /username:MyUserName
/password:MyPassword
```

SQL Packager sets two return codes that you can use in batch files or scripts to determine whether the package ran successfully:

- **0** is returned if the package ran successfully
- **-1** is returned if an error occurred

If you have specified `/quiet` to run without the graphical user interface, the return code is stored in the ERRORLEVEL environment variable, and the error message is written to the console.

Understanding the results

This topic provides information that may help you to understand the results when you use SQL Packager to create or upgrade a database. You may also wish to refer to Troubleshooting.

Database diagrams

SQL Packager does not package or upgrade database diagrams.

System tables

SQL Packager does not package or upgrade system tables.

Encrypted database objects

If you are packaging a SQL Server 2000 database that contains an encrypted user-defined function, stored procedure, trigger, or view and you have system administrator permissions, SQL Packager decrypts the object and you can view its internal SQL in the schema packaging script. If you do not have system administrator privileges, you cannot package the encrypted object.

SQL Packager cannot decrypt views, stored procedures, functions, and DML triggers that are encrypted on a SQL Server 2008 or SQL Server 2005 database. Therefore, SQL Packager cannot display the SQL code for the encrypted objects, and cannot package them.

Column order

If you are upgrading a database, column order is not forced unless you select the **Force table column order to be identical** schema packaging option.

For example, the latest version database has a table that contains *ColA* and *ColB*, in that order, and the previous version database has the same table but with *ColB* then *ColA*. If **Force table column order to be identical** is not selected, SQL Packager considers the tables to be identical. If the option is selected, SQL Packager considers the tables to be different and upgrades the table.

Renamed columns

If you are upgrading a database, SQL Packager attempts to recognize renamed columns by the similarity of the names and the data types of the columns. When a renamed column is recognized as such, SQL Packager renames the column as appropriate.

However, if the names and data types are very different, SQL Packager may consider the renamed column to be a completely different column. In this case, if *ColA* in the latest version database is renamed to *ColB* in the previous version database, when SQL Packager creates the upgrade script, *ColA* will be created in the previous version database as a new column and *ColB* will be deleted. To avoid data loss, before you run the package you must take care to preserve any data in the two columns, and merge them following the upgrade.

Updated views

Following an upgrade, if a view has not been updated by the package and it contains a `SELECT *` statement, you must refresh it using **sp_refreshview**, to reflect any changes that have been made to the underlying objects on which the view depends. Refer to your SQL Server documentation for more information.

It is not best practice to use `SELECT *` statements in views; you are recommended to specify an explicit column list.

Replication

If objects that are used in replication are upgraded, errors may occur. For example, SQL Packager cannot drop a table if it is used for replication.

Users

In Microsoft Windows, users are a composite of the domain name or computer name and the user name, for example *Computer1\WindowsUser1*. If you are upgrading a database, SQL Packager references only the user name, so that *Computer1\User1* and *Computer2\User1* would be considered as the same. Therefore, if you intend to upgrade users, ensure that their user names are different.

SQL Packager upgrades changes to users, such as changes to permissions. However, SQL Packager does not upgrade modifications to user passwords.

New users are created with the password: **p@ssw0rd**.

Filegroups

SQL Packager supports the upgrade of databases that use multiple filegroups. However, you must ensure that the filegroups have been created on the target server prior to upgrade. If the filegroups do not exist, the upgrade will fail.

CLR assemblies

When a CLR assembly is to be updated, if possible SQL Packager achieves this by using ALTER ASSEMBLY.

If SQL Packager determines that it would not be possible to use ALTER ASSEMBLY, the relevant table is rebuilt twice:

- in the first rebuild, the CLR type columns are converted to *nvarchar*
  The CLR type columns are dropped and recreated.
- in the second rebuild, the *nvarchar* data is converted to the final CLR type

Data is preserved.

> The ToString representation of the CLR user-defined type must be the same for both the old and the new assembly, otherwise the upgrade may fail, or the data may be corrupted.

To force SQL Packager to use the double-table rebuild method, select the schema packaging option. Do not use ALTER ASSEMBLY to change CLR types.

Partition schemes and functions

In SQL Server 2008 and SQL Server 2005, partition schemes can be specified for tables so that the table is stored in several filegroups. By default, SQL Packager ignores filegroups. However, if you select the schema packaging option Consider next filegroups in partition schemes, SQL Packager upgrades the files.

> For updates to partition schemes, a large amount of disk space may be required on the defined filegroups because partition ranges must be merged and split.

In certain cases, for example when a partition function changes from left range to right range, it is necessary to drop and recreate partition functions and partition schemes. In these cases, the table is rebuilt twice:

- in the first table rebuild, the content is saved to a temporary filegroup
- in the second table rebuild, the table is migrated from the temporary filegroup to a new partition scheme

Data is preserved.

> If a CLR assembly upgrade also requires a table to be rebuilt twice, the CLR assembly and the partition scheme are upgraded at the same time.

Certificates, symmetric keys, and asymmetric keys

SQL Server 2005 severely restricts access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Packager can package only the permissions of certificates and asymmetric keys; symmetric keys cannot be packaged. To ignore all certificates, symmetric keys, and asymmetric keys, select the Ignore certificates, symmetric and asymmetric keys schema packaging option.

Extended properties on databases

Extended properties on databases are always packaged if they differ. If you do not want them to be packaged, select the Ignore extended properties schema packaging option.

Setting packaging options

The packaging options are a set of advanced features that enable you to modify the behavior of SQL Packager. For example, you can set SQL Packager so that it ignores certain features, such as triggers.

To display the **SQL Packager Options** dialog box:

- On the **Packager** menu, click **Options**.

The options are divided into the following tabs:

- **Schema options** apply to the structure of the database
- **Data options** apply to the data for the selected objects
- **Packager options** apply to the .EXE or C# project

The packaging options are saved for each user. Therefore, if you change the options they will apply to all projects run by the current user.

To reset all the options to their default values, click **Restore All Defaults**.

Setting schema packaging options

The schema packaging options are a set of advanced features that enable you to modify the behavior of SQL Packager when it packages the database structure. For example, you can set SQL Packager so that it ignores certain objects, or so that it does not script certain properties in the package (such as the collation order on columns).

Treat items as case sensitive

For databases with case-sensitive collation, enables objects with case-sensitive names to be packaged. For example, considers object names such as *ATable* and *atable* as different and performs case-sensitive comparisons on stored procedures, and so on.

You should use this option only if you have databases with binary sort order or case-sensitive sort order.

Force table column order to be identical

If additional columns are inserted into the middle of a table, this option forces a rebuild of the table so the column order is correct following upgrade. Data will be preserved.

Do not include plumbing for transactional synchronization scripts

Removes transactions from the package to produce SQL code that is more readable.

If this option is not selected and the package fails, the script is rolled back to the start of the failed transaction. If this option is selected, the script is not rolled back. This can be useful for detection of errors within a script.

Add WITH ENCRYPTION option to stored procedures etc

Adds `WITH ENCRYPTION` when stored procedures, functions, views, and triggers are included in the package.

Note that if you use `ADD ENCRYPTION` on a SQL Server 2008 or SQL Server 2005 database, SQL Packager will not subsequently be able to display, or package the encrypted objects.

Do not use ALTER ASSEMBLY to change CLR types

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

If CLR types are to be packaged, this option forces two rebuilds of the table with conversion to and from strings to update the CLR types, instead of using `ALTER ASSEMBLY`. For a detailed explanation, see Understanding the Results.

Consider next filegroups in partition schemes

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

When this option is selected, if a partition scheme contains a next filegroup, SQL Packager considers the next filegroup for the upgrade if the filegroup is extended. The next filegroup does not affect the way in which data is stored. For a detailed explanation, see Understanding the Results.

To ignore next filegroups, clear the check box.

Disable and later re-enable DDL triggers

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

DDL triggers can cause problems when you run the packaging script. Select this option to disable any enabled DDL triggers before upgrading the databases, and re-enable those triggers on completion.

Include dependencies

Include dependencies is only available in SQL Packager 6.4 and later

When this option is selected, SQL Packager checks for object dependencies; if you excluded objects and other objects that you selected are dependent on the excluded objects, the excluded objects are packaged. For example, if you select a stored procedure and it references a table, even if you excluded that table, the table is still packaged.

By default, SQL Packager will include dependencies in the package. Clear the option if you do not want to include the dependencies. Note that clearing this check box may produce unexpected results or the script may fail. Roles and users are always included in the package.

Decrypt encrypted objects in 2008 and 2005 databases

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

When this option is selected, SQL Packager decrypts text objects in SQL Server 2008 and SQL Server 2005 databases which were created using the WITH ENCRYPTION option.

This option can have a significant performance impact on large databases.

When SQL Packager saves a snapshot, this option is set, and all encrypted objects are decrypted.

Ignore indexes

Ignores indexes, unique constraints, and primary keys when packaging the database structure.

Ignore permissions

Ignores permissions on objects when packaging the database structure.

Ignore DML triggers

Ignores DML triggers when packaging the database structure.

Ignore constraint and index names

Ignores the names of indexes, foreign keys, primary keys, and default, unique, and check constraints when displaying the objects that are available for packaging.

> Note that they will be included in the schema packaging script.

Ignore white space

Ignores white space (newlines, tabs, spaces, and so on) when displaying the objects available for packaging.

> White space will not be ignored when the objects are created or upgraded.

Ignore comments

Ignores comments when comparing views, stored procedures, and so on.

> Comments will be included in the schema packaging script.

Ignore full text indexing

Ignores full-text indexes, catalogs, and so on when packaging databases.

Ignore users' permissions and role memberships

When role-based security is used, object permissions are assigned to roles, not users. If this option is selected, SQL Packager creates or upgrades object permissions only for roles, and members of roles that are roles. Users' permissions and role memberships are ignored.

Ignore statistics

Ignores statistics when packaging the database structure.

Ignore foreign keys

Ignores foreign keys when packaging the database structure.

Ignore check constraints

Ignores check constraints when packaging the database structure.

Ignore identity seed and increment values

For identity properties, ignores only the identity seed and increment values when displaying the objects that are available for packaging.

> Note that they will be included in the schema packaging script.

Ignore fill factor and index padding

Ignores the fill factor and index padding in indexes and primary keys when packaging the database structure.

Ignore INSTEAD OF triggers

Ignores INSTEAD OF DML triggers when packaging the database structure.

Ignore bindings

Ignores bindings on columns and user-defined types. For example, **sp_bindrule** and **sp_bindefault** clauses will not be included in the schema packaging script.

Ignore WITH NOCHECK on foreign keys and check constraints

Ignores the WITH NOCHECK argument on foreign keys and check constraints.

> Foreign keys or constraints that are *disabled* are not ignored.

Ignore filegroups, partition schemes and partition functions

Ignores filegroup clauses, partition schemes, and partition functions on tables and keys when packaging the database structure. Partition schemes and partition functions are not available for inclusion in the package when this option is selected.

Ignore extended properties

Ignores extended properties on objects and databases when packaging databases.

Ignore SET QUOTED_IDENTIFIER and SET ANSI_NULLS statements

Ignores these SET statements when displaying available views, stored procedures and so on.

> Note that these statements will be included in the schema packaging script.

Ignore collation order

Ignores collation order on character data type columns when packaging databases.

Ignore certificates, symmetric and asymmetric keys

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

SQL Server 2005 severely restricts access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Packager can package only the permissions of certificates and asymmetric keys; symmetric keys cannot be packaged.

Ignore trigger order

DML triggers can have an order specified, such as `FIRST INSERT, LAST UPDATE`, and so on. Select this option to ignore the trigger order for DML triggers when packaging databases. Note that the DDL trigger order is not affected.

Ignore event notifications on queues

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

Ignores the event notification on queues when packaging databases.

Ignore users' properties in comparison

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

If this option is not selected, SQL Packager compares user properties, such as the type of user (SQL, Windows, certificate-based, asymmetric key based) and any schema to identify differences. If a user is selected to be upgraded, SQL Packager upgrades the properties where possible.

If you select this option, users' properties are ignored, and only the user name is packaged.

Ignore the order of WITH elements

If a stored procedure, user-defined function, DDL trigger, DML trigger, or view contains multiple WITH elements (such as encryption, schema binding, and so on), select this option to ignore the order of the `WITH` elements when packaging databases.

Ignore the lock properties of indexes

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

Ignores index `PAGE LOCK` and `ROW LOCK` properties when packaging databases.

Ignore replication triggers

Ignores replication triggers when packaging databases.

Ignore identity properties

Ignores the identity property on columns when displaying the objects that are available for packaging.

> Note that the identity property will be included in the schema packaging script.

Setting data packaging options

The data packaging options are a set of advanced features that enable you to modify the behavior of SQL Packager when it packages data. For example, you can set options so that triggers are disabled when you upgrade a database and then, when the upgrade is complete, re-enable the triggers.

> You can also set:
>
> - Schema options for packaging the structure of the database
> - Packager options for the .EXE or C# project

Disable foreign keys

Disables foreign keys before creating or upgrading the database, and re-enables them on completion.

> In some circumstances, foreign keys will be dropped and recreated rather than disabled and re-enabled.

Drop primary keys, indexes and unique constraints

Drops then recreates primary keys, indexes (including XML indexes and partitioned indexes), and unique constraints before creating or upgrading the database, and re-enables them on completion. If the primary key, index, or unique constraint is the comparison key, it cannot be dropped.

Select this option to ensure that unique constraints are not violated when data is inserted into tables or modified. When the constraints are re-created, the data is verified to ensure that no constraints have been violated. (If they have, the script will fail.)

Do not use transactions in synchronization SQL scripts

Removes transactions from the synchronization SQL scripts to produce SQL code that is more readable.

If this option is not selected and the synchronization script fails, the script is rolled back to the start of the failed transaction. If this option is selected, the script is not rolled back. This can be useful for detection of errors within a script.

Transport CLR types as binary

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

If this option is selected, SQL Packager uses the binary representation of CLR types; if this option is not selected, SQL Packager uses the string representation.

> String representations do not always contain the full information about the data.

Disable DML triggers

Disables DML triggers before creating or upgrading the database, and re-enables them on completion.

For example, you may want to disable triggers if you have a trigger defined on a table that inserts data into another table on `INSERT`, `DELETE`, or `UPDATE`; if you do not, the data in the tables will change as the package is run, which will cause unpredictable results.

Disable DDL triggers

This option is used only for SQL Server 2008 and SQL Server 2005 databases.

Disables DDL triggers before creating or upgrading the database, and re-enables them on completion.

Reseed identity columns

Reseeds identity values in the database you are updating after synchronization.

Trim trailing spaces

If the data in two columns differs only by the number of spaces at the end of the string, SQL Packager considers the data to be identical. This option does not apply to CLR or XML columns.

If this option is selected, trailing spaces are ignored when creating or upgrading databases.

Force binary collation

SQL Packager uses keys to compare rows. If the comparison key is a string, this option forces a binary collation on all string sorting.

Use checksum comparison

Performs a checksum prior to comparison. The data is compared only if the checksums differ. Note that if the data differs only in text or image columns, the checksums will be identical and SQL Packager will consider the data to be identical.

For SQL Server 2000, db_owner permissions are required.

Ignore case

When you are upgrading a database, SQL Packager will ignore the case of the object names, if this option is selected. For example, SQL Packager will consider [dbo].[Widget] to be the same as [dbo].[wIDgEt] and will compare the data in the two tables.

> If the databases that you are upgrading are running on a SQL Server that uses case-sensitive sort order, you should ensure that this option is cleared.

Ignore spaces

When you are upgrading a database, SQL Packager will ignore spaces in object names, if this option is selected. For example, SQL Packager will consider [dbo].[Widget Prices] to be the same as [dbo].[WidgetPrices] and will compare the data in the two tables.

Ignore underscores

When you are upgrading a database, SQL Packager will ignore underscores in object names, if this option is selected. For example, SQL Packager will consider [dbo].[Widget_Prices] to be the same as [dbo].[WidgetPrices] and will compare the data in the two tables.

Include views

If this option is selected, SQL Packager includes views in the upgrade. Generally, views can be updated only if the referenced rows are from a single table, and the referenced columns are simple (for example, they must not include identity columns or computed columns).

Include identity columns

Includes identity columns in the package. Note that if you do not select this option and an identity column is used as the primary key for a table, it will be included in the packaging script.

Include timestamp columns

If this option is selected, SQL Packager will compare timestamp columns when you are upgrading a database.

> Timestamp columns cannot be included in the data packaging script.

Setting SQL Packager options

The packager options enable you to specify default settings for your packages. SQL Packager uses the packager options when you create an .EXE or a C# project.

You can also set:

- schema options for packaging the structure of the database
- data options for packaging the data

Automatically increment package name (if already exists)

If an executable file or project file with the same name already exists at the default package location, SQL Packager will automatically increment the package name when this option is selected.

Default package location

You use this setting to specify the default location for your package files. Type or select the path for the **Default package location**, or click the browse button to choose the folder.

Maximum transaction size

You can use this box to specify the amount of data included in each transaction.

Using the command line

The command line interface provides access to the functionality provided by SQL Packager. For example, using the command line interface you can:

- automate the comparison, synchronization, and packaging of both database structures and data
- perform scheduled comparisons and synchronizations
- synchronize multiple databases
- upgrade customer databases without manual intervention

You invoke the command line either from a script, such as a batch script or VBScript, or by using the facilities provided by compiled languages such as C++ and C#.

These pages provide a description of basic command line features and examples illustrating how you can use the command line interface.

Getting help from the command line

To display full help on all of the switches that are available for the command line, at the command prompt enter:

```
sqlpackager /help /verbose
```

where `/help` displays the help message and can be used in conjunction with `/verbose` for more detailed information. To output the help in HTML format, use the `/html` switch, for example:

```
sqlpackager /help /verbose /html > filename.htm
```

If you specify any other switches, they are ignored.

Prerequisites

To use the SQL Packager command line interface, you must have:

- a license for SQL Packager 6.0 (or later versions), a license for the Professional Edition of SQL Packager version 5.5 and earlier, or a SQL Toolbelt license
  If you do not have a license, you can use the command line for 14 days.
- .NET framework version 2.0 or later
  This is required to run the command line interface, but it is not required when you develop applications and scripts that use the interface.
- MDAC 2.8 or later

For information about distributing the command line interface with your application, see Integrating the command line with applications.

Basic command line features

This topic describes how to use the basic features of the command line.

Entering a command

When you enter a command line, the order of switches is unimportant. You are recommended to follow the Microsoft convention of separating a switch from its values using a colon as shown below.

```
/out:output.txt
```

(You can separate a switch that accepts a single value from its value using a space, but this is not recommended.) Values that include spaces must be delimited by double quotation marks (`"`). For example:

```
/out:"c:\output file.txt"
```

if you delimit a path with double quotation marks, you must not terminate the path with the backslash character (`\`), because the backslash will be interpreted as an escape character. For example:

Incorrect: `/location:"C:\Packages\"`

Correct: `/location:"C:\Packages"`

For switches that accept multiple values, use commas to separate the values. For example:

```
/options:IncludeDependencies,ForceColumnOrder
```

For switches that accept a compound value, separate each part of the value using a colon. For example, the `/includeschema` and `/excludeschema` switches are used to include and exclude database objects from the actions performed by the tool. For example:

```
/includeschema:table:Product
```

includes all tables for which the table name contains the word *Product*.

Aliases

Many of the switches have an alias. The alias provides a convenient short-hand way to specify the switch. For example, `/?` is the alias for the `/help` switch, and `/v` is the alias for the `/verbose` switch. Note that switches and aliases are not case-sensitive.

/options switch

You can use the `/options` switch to change your options. For example, by default, comparisons are not case-sensitive; to specify case-sensitive comparisons:

```
/options:CaseSensitiveObjectDefinition
```

If you set any options explicitly, all of the default options are switched off.

Refer to the full command line help for more information about which options are set by default, and all the options that are available.

`/options` can't be used with a project file - you need to set the options in the project file instead.

/verbose and /quiet switches

The standard output mode prints basic information about what the tool is doing while it is executing. You can specify verbose and quiet modes using the */verbose* and */quiet* switches, respectively: in verbose mode, detailed output is printed; in quiet mode, output is printed only if an error occurs.

Redirecting command output

Output from all commands can be redirected to a file by one of several methods:

- Use the `/out` switch to specify the file to which you want output directed:
  `sqlpackager ... /out:outputlog.txt` where `outputlog.txt` is the name of the file. If the file exists already, you must also use the `/force` switch to force the tool to overwrite the file, otherwise an error will occur.
- Use the output redirection features that are provided by the shell in which you are executing the command.
  From the standard command prompt provided by Windows, you can redirect output to a file as follows:
  `sqlpackager ... > outputlog.txt`

  The redirection operator (>) and file name must be the last items on the command line.

  If the specified file exists already, it will be overwritten. To append output from the tool to an existing file, for example to append to a log without losing the data already present in the log, enter the following:
  `sqlpackager ... >> existinglog.txt`
  If you are scripting using a language such as VBScript, JScript, PHP, Perl, or Python, or if you want to access the tool from Web pages using ASP.NET, refer to the documentation for the language.
- Specify command line arguments in an XML file that can be referenced using the `/argfile` switch.
  For details, see Using XML to specify command line arguments.

Examples using the command line

This topic provides some simple examples of how to use the command line interface.

You may also wish to refer to Frequently asked questions for the command line.

Packaging databases

To package database *WidgetSales* on the local SQL Server, creating a package executable called *WidgetPackage.exe* in *C:\Packages*, which will create a new, identical database:

```
sqlpackager /database1:WidgetSales /location:"C:\Packages"
     /name:WidgetPackage /makeexe
```

To create the same package and run it immediately (for example, if you want to test the package):

```
sqlpackager /database1:WidgetSales /location:"C:\Packages"
     /name:WidgetPackage /makeexe /run
```

To package database *WidgetSales* on the local SQL Server, creating a C# project called *WidgetPackage* in *C:\Packages\Projects*, which will create a new, identical database:

```
sqlpackager /database1:WidgetSales /location:"C:\Packages\Projects"
     /name:WidgetPackage /makeproject
```

To create the same package and open it immediately in Microsoft Visual Studio:

```
sqlpackager /database1:WidgetSales /location:"C:\Packages\Projects"
     /name:WidgetPackage /makeproject /open
```

To package an upgrade from *WidgetDev* to *WidgetLive* creating a package executable called *WidgetPackage.exe* in *C:\Packages*:

```
sqlpackager /database1:WidgetDev /database2:WidgetLive\\     /location:"C:\Packages"
/name:WidgetPackage /makeexe
```

In this example, when the package is run, *WidgetLive* will be updated.

Using */presql* and */postsql*

Packages created by SQL Packager support the `/presql` and `/postsql` switches. These switches allow you to specify SQL scripts to run before and after the package executes.

> Note that any SQL scripts specified by `/postsql` are always run, even if errors occur when the package executes.

For example, to run the script *WidgetPostScript.sql* after the package *WidgetPackage.exe*, enter:

```
widgetpackage.exe /postsql:WidgetPostScript.sql
```

Frequently asked questions for the command line

Licensing

How do I activate the command line tools?

To use the SQL Packager command line interface, you must have one of the following licenses:

- SQL Packager 6.0 (or later)
- SQL Packager Professional Edition 5.5 (or earlier)
- SQL Toolbelt

You can also redistribute your application if you have a valid license. For more information, see Integrating the command line with applications.

Comparing databases

How can I include or exclude the object definition for specific tables?

You can use the `/includeschema` or `/excludeschema` switch with regular expressions to do this.

How can I include or exclude the data for specific tables?

You can use the `/includedata` or `/excludedata` switch with regular expressions to do this.

What is included or excluded when I use a project?

When you use a project, all objects and data that were selected for inclusion when the project was saved are automatically included; you do not need to explicitly include them using the `/includeschema` or `/includedata` switches. You can override the inclusion by specifying the `/excludeschema` or `/excludedata` switches as required.

Integration

How do I integrate the command line tools with applications?

For information about how to integrate the command line tools with applications that you distribute to your customers, see Integrating the command line with applications.

How do I integrate database package creation with a build process?

You may want to install a package as part of your application installation process. You can create the package during the build, bundle it as part of your installer, and set up the installer to execute it at the appropriate point during the installation.

The build script example below shows how you would execute the SQL Packager command line tool from an NAnt build script. You can modify this task for use with other build systems such as Visual Build.

```
... <!-- Use your SQL command line installation directory --> <property
name="sqlCmdLineInstallDir" value="..."/> ... <target name="PackageDatabase"
description="Create executable database package."> <exec
basedir="${sqlCmdLineInstallDir}" program="sqlpackager.exe"
commandline="/database1:FirstDatabaseName /makeexe /name:PackageName
/location:TargetDirectoryName"/> </target> ...
```

By default, NAnt captures the output of the program and incorporates it as part of the build log. Alternatively, you can use the *output* attribute of the *exec* task to specify a file to which the output is to be redirected.

Integrating the command line with applications

To integrate the SQL Packager command line tool with applications that you distribute to your customers, you must have one of the following licenses:

- SQL Packager 6.0 (or later)
- SQL Packager Professional Edition 5.5 (or earlier)
- SQL Toolbelt

When you have a license, and you execute the tool, the distribution files that you need to distribute the applications are generated; these files are marked with an asterisk (*) below.

The files that you should bundle into your application installer are listed below. The files should be installed in the same folder in which your application is installed.

Note that to distribute a package created with the SQL Packager command line, you need to distribute only the package executable.

- SQLPackager.exe
- RedGate.CommandLine.Common.dll
- RedGate.SQLPackager.Distribution.dll*
- RedGate.SQLPackager.Distribution.mod*
- RedGate.Shared.SQL.dll
- RedGate.Shared.Utils.dll
- RedGate.SQlCompare.ASTParser.dll
- RedGate.SQLCompare.Rewriter.dll
- RedGate.SQLCompare.Engine.dll
- RedGate.SQLCompare.CommandLine.dll
- RedGate.SQLDataCompare.Engine.dll
- RedGate.SQLDataCompare.CommandLine.dll
- RedGate.SQLPackager.Engine.dll
- RedGate.SQLPackager.CommandLine.dll
- SQLPackager.exe.config
- RedGate.Compression.ZLib.dll
- SQL Packager Code Templates (folder)

Using XML to specify command line arguments

You can use an XML file to specify the arguments for the command line interface. You may want to do this because:

- An XML file is easier to read than a long and complex command line, particularly where complex rules for including and excluding objects are specified.
- You can easily transform an XML file into other formats using XSLT.
  For example, you could transform your argument file to HTML for presentation on a Web page.
- Using an XML file overcomes some limitations that can be a problem when you want to specify regular expressions as command line arguments.
  For example, you may want to use the pipe character (|) as part of a regular expression, but it causes problems when it is used at the command prompt; if you use an XML file you can use the pipe character with no problems.
- Most programming languages support XML, through built-in or freely available third-party libraries.
- This makes it easy to generate and process the XML file.

Create the XML file in the following format:

```
<?xml version="1.0"?>\\<commandline>
<switch_name1/>
<switch_name2>switch_value</switch_name2> ....
</commandline>
```

For example, for the `/includeschema` and `/excludeschema` switches, use the following format:

```
<includeschema>objecttype:RegularExpression</includeschema>
```

To execute the command line tools using an XML argument file as input, at the command prompt enter:

```
sqlpackager /argfile:XMLfilename.xml
```

When you use an XML file to supply the arguments, you cannot specify any other switches on the command line except `/verbose` or `/quiet`.

Worked examples

These detailed examples demonstrate how to complete tasks using SQL Packager

- Worked example: Packaging a database as an .EXE
  This worked example demonstrates how to package a database as an .EXE (.NET executable) and run the executable to create a copy of the database.
- Worked example: Packaging an upgrade as a C# project
  This worked example demonstrates how to package an upgrade to a database as a C# project, and run the package.

The worked examples should take you no more than 10 minutes each to complete.

Packaging a database as a .NET executable

This worked example demonstrates how to package a database as an .EXE (.NET executable) and run the executable to create a copy of the database.

In the example, the Magic Widget Company has a SQL Server database running on a live Web server. They have created a database of their products, which now needs to be packaged for deployment to the sales department.

You will see how to:

1. Set up the database if you want to follow the example on your own system.
   You will need access to a SQL Server to do this.
2. Specify the contents of the package.
3. Preview the SQL scripts.
4. Generate the package as an .EXE (.NET executable).
5. Run the package to create a copy of the database.

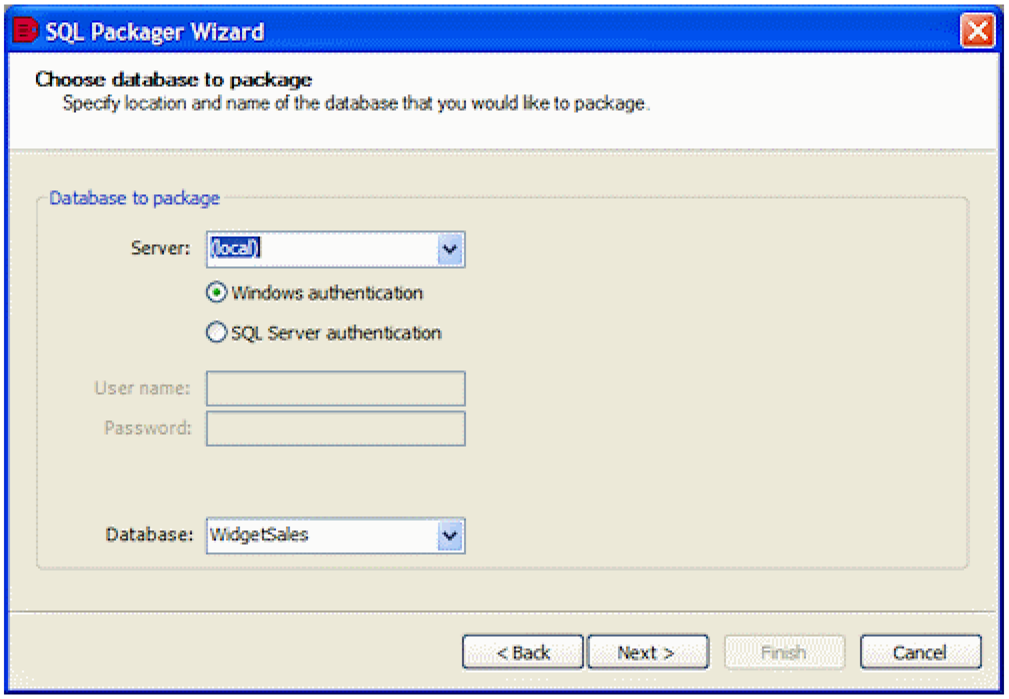
Setting up the database

The worked example packages the *WidgetSales* database. To create this database on your SQL Server:

1. If it exists already, delete the database *WidgetSales* from your SQL Server.
2. Click here to view the SQL creation script for the database.
3. Copy the script, paste it in your SQL application, and then run it.
   The database is created and populated with data.

Specifying the package contents

1. If you have not yet started SQL Packager, select it from your **Start** menu; if SQL Packager is already running, click
   New Project.
2. On the **Choose a project type** page of the Packager Wizard, select **Package a database**, and click **Next**.
   The **Choose database to package** page of the Packager Wizard is displayed.
3. In the **Server** box, under **Database to package**, type or select the name of the SQL Server on which you created the database.
4. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
5. In the **Database** box, type or select *WidgetSales*.
   If *WidgetSales* is not displayed in the **Database** list, right-click in the **Database** box and click **Refresh**, or scroll to the top of the list and click **Refresh.**
   .



SQL Packager Wizard

Choose database to package
Specify location and name of the database that you would like to package.

Database to package

Server: (local)

[✓ Windows authentication]

SQL Server authentication

User name:

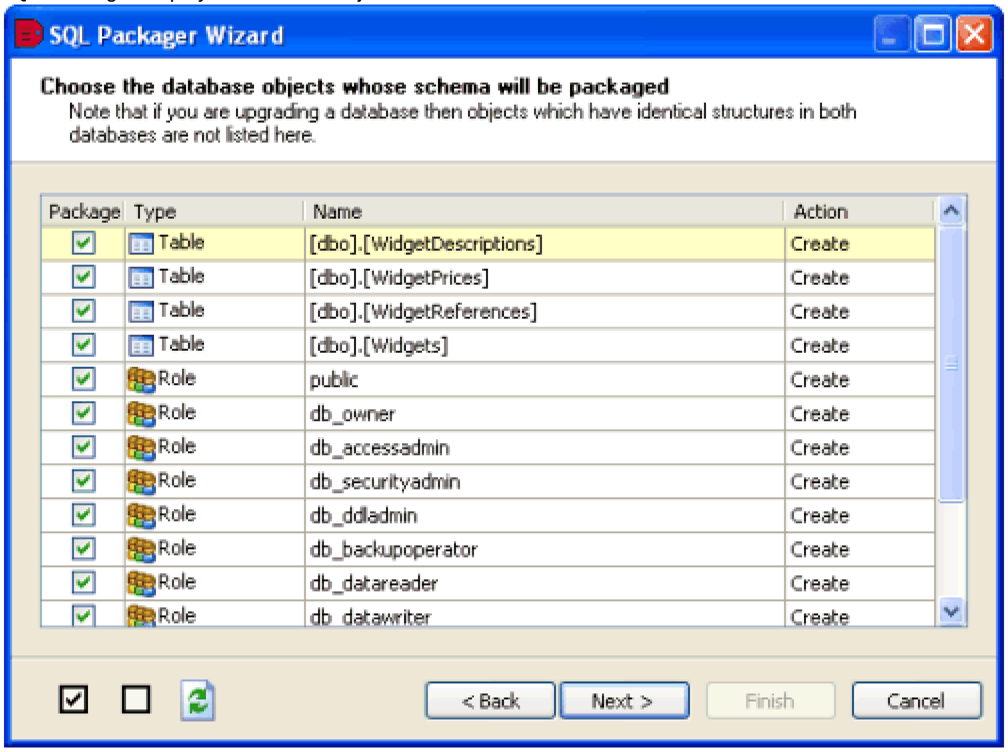
Password:

Database: WidgetSales

6. Click **Next**.
   SQL Packager displays a message dialog box while it analyzes the database structure.

If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that you choose the database. For this example, leave the setting as it is.

7. Click **OK** to close the message box.
   SQL Packager displays a list of the objects in the database.

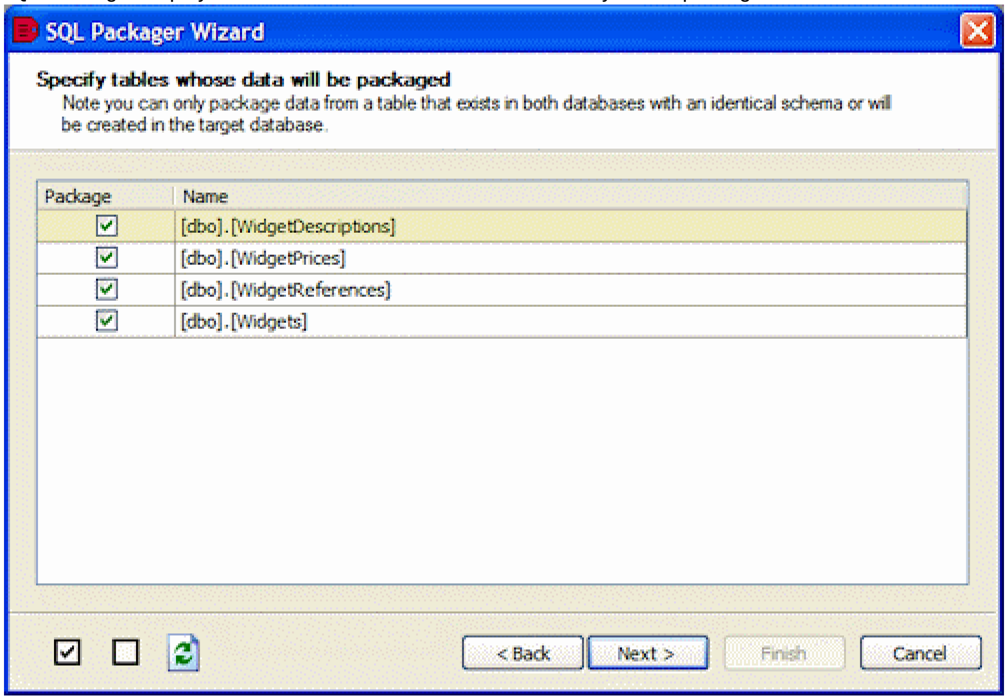


Choose the database objects whose schema will be packaged
Note that if you are upgrading a database then objects which have identical structures in both databases are not listed here.

Package	Type	Name	Action
✓	Table	[dbo].[WidgetDescriptions]	Create
✓	Table	[dbo].[WidgetPrices]	Create
✓	Table	[dbo].[WidgetReferences]	Create
✓	Table	[dbo].[Widgets]	Create
✓	Role	public	Create
✓	Role	db_owner	Create
✓	Role	db_accessadmin	Create
✓	Role	db_securityadmin	Create
✓	Role	db_ddladmin	Create
✓	Role	db_backupoperator	Create
✓	Role	db_datareader	Create
✓	Role	db_datawriter	Create

You can choose which objects to package. For more information, see Specifying the package contents.
For this example, leave all the check boxes selected so that all of the objects are created in the database when the package is run.

8. Click **Next**.
   SQL Packager displays a list of the tables that contain data that you can package.



Specify tables whose data will be packaged
Note you can only package data from a table that exists in both databases with an identical schema or will be created in the target database.

Package	Name
✓	[dbo].[WidgetDescriptions]
✓	[dbo].[WidgetPrices]
✓	[dbo].[WidgetReferences]
✓	[dbo].[Widgets]

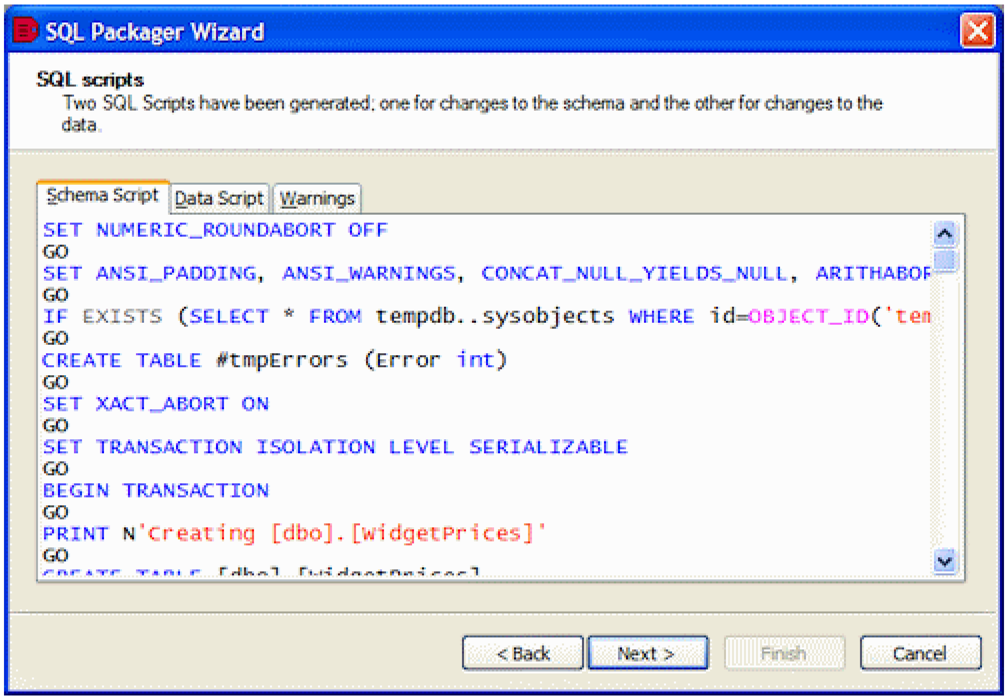
You can choose the tables for which you want to package data. For this example, leave all the check boxes selected so that you package all of the data.

9. Click **Next**.
   SQL Packager displays a message dialog box while it generates the SQL script.
   If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that it generates the SQL script. For this example, leave the setting as it is.

10. Click **OK** to close the message box.

Previewing the SQL Scripts

The **SQL scripts** page is displayed.



SQL scripts
Two SQL Scripts have been generated; one for changes to the schema and the other for changes to the data.

```
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABOR
GO
IF EXISTS (SELECT * FROM tempdb..sysobjects WHERE id=OBJECT_ID('tem
GO
CREATE TABLE #tmpErrors (Error int)
GO
SET XACT_ABORT ON
GO
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
GO
BEGIN TRANSACTION
GO
PRINT N'Creating [dbo].[WidgetPrices]'
GO
```

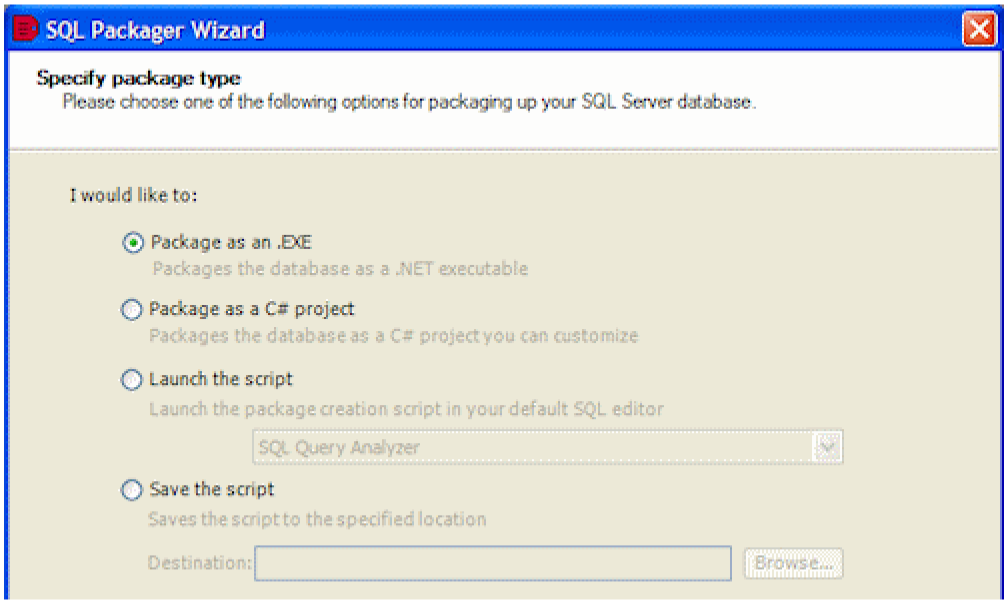
To see the SQL script for creating the data, click the **Data Script** tab. To see details about unexpected behavior that may occur when you run the package, click the **Warnings** tab.

If required, you can save the scripts on the next page of the wizard.

When you have finished reviewing the SQL scripts, click **Next**.

Generating the package

The **Specify package type** page is displayed.



Specify package type
Please choose one of the following options for packaging up your SQL Server database.

I would like to:

[✓ Package as an .EXE]
Packages the database as a .NET executable

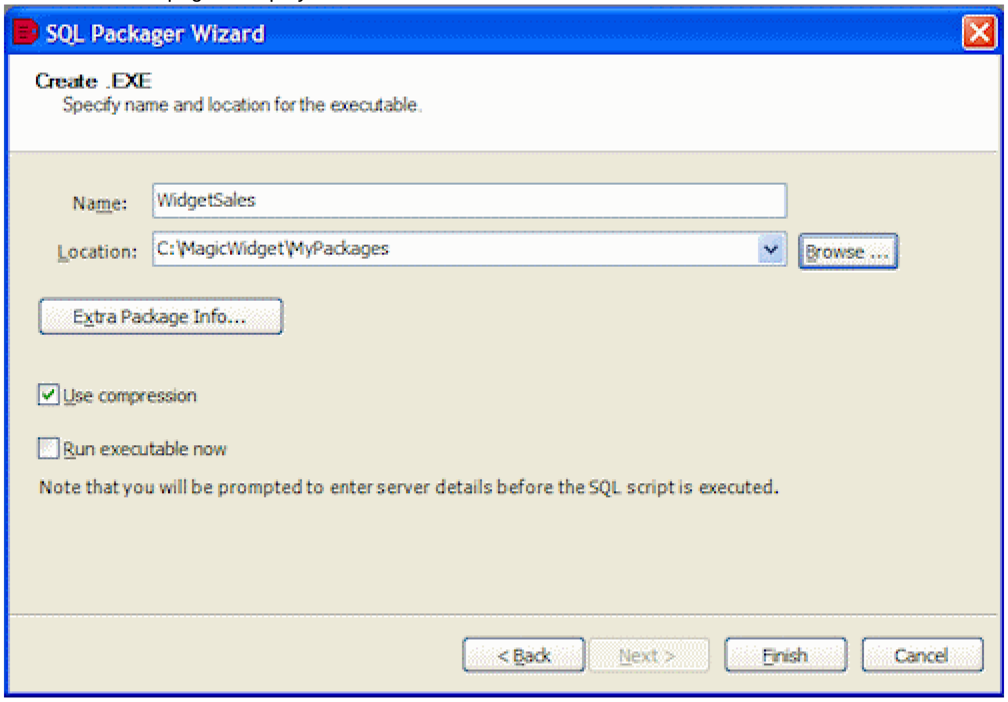
Package as a C# project
Packages the database as a C# project you can customize

Launch the script
Launch the package creation script in your default SQL editor

Save the script
Saves the script to the specified location

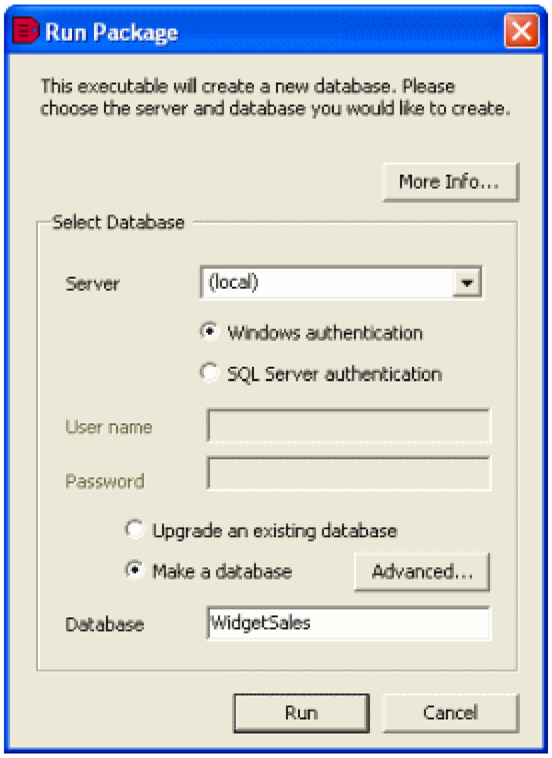
Destination:

In this example, we will create an .EXE. For an example of how to create a C# project, see Packaging an upgrade as a C# project.

1. Ensure that **Package as an .EXE** is selected, and click **Next**.
   The **Create .EXE** page is displayed.
   
   SQL Packager Wizard
Create .EXE
Specify name and location for the executable.
Name: WidgetSales
Location: C:\MagicWidget\MyPackages
Browse ...
Extra Package Info...
Use compression
Run executable now
Note that you will be prompted to enter server details before the SQL script is executed.
< Back
Next >
Finish
Cancel
2. Enter a name and location for your package.
3. Leave the **Use compression** check box selected so that your package will be compressed.
   The generated files will be compressed to approximately 75% of their original size.
4. Select the **Run executable now** check box, and click **Finish**.
   A message dialog box informs you that the executable is created in the location you specified. Click **OK** to close it. For large databases, additional dynamic-link library (.dll) files are also created. The **Run Package** dialog box is displayed for you to run the executable immediately.

Running the package

You use the **Run Package** dialog box to specify details of the database that will be created when the package is run.


Run Package
This executable will create a new database. Please choose the server and database you would like to create.
More Info...
Select Database
Server (local)
Windows authentication
SQL Server authentication
User name
Password
Upgrade an existing database
Make a database
Advanced...
Database WidgetSales
Run
Cancel

1. Select the **Server** on which you want to create the database, and if required, enter the authentication details.

   > The SQL Server version must be compatible with the latest version database that you specified when you chose the database to package.

2. The **Advanced** options enable you to define properties for the database that will be created, such as the database location.
3. Type a name for the **Database** and click **Run**.
4. A message dialog box is displayed for you to confirm that you want to continue. Click **Yes**.
5. When the database is created, a message dialog box confirms that the package has run successfully. Click **OK** to close it.

You can use your SQL application to check that the database has been created as you expect. If you have purchased Red Gate **SQL Compare** y ou can compare the databases' structure to confirm that they are identical; if you have purchased **SQL Data Compare**, you can compare the data to confirm it is identical.

Packaging an upgrade as a C# project

This worked example demonstrates how to package an upgrade to a database as a C# project, and run the package.

In the example, the sales department of the Magic Widget Company has a SQL Server database of their products. The development department have made a number of changes to the structure and content of the product database, which now need to be packaged for deployment to the sales department as an upgrade.

You will see how to:

1. Set up the databases if you want to follow the example on your own system.
   You will need access to a SQL Server to do this.
2. Specify the contents of the package.
3. Preview the SQL scripts.
4. Generate the package as a C# project.
5. Run the package to upgrade the existing database.
   You will need Microsoft® Visual Studio® .NET 2005 or later to compile the project.

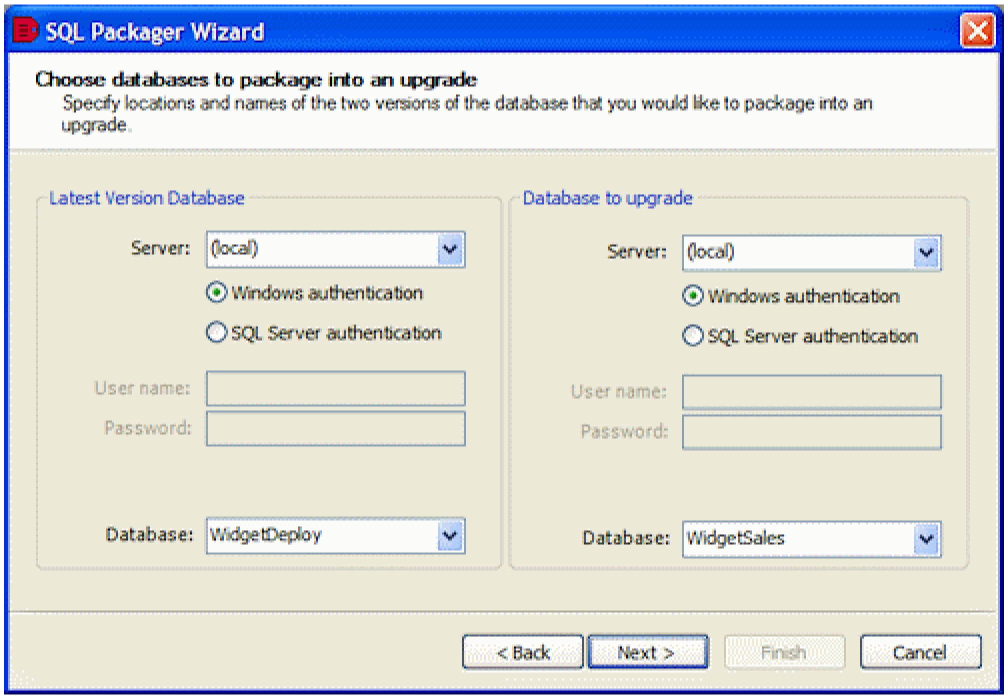
Setting up the databases

The worked example upgrades the product database, *WidgetSales*, with a later version of the database called *WidgetDeploy*. To create these databases on your SQL Server:

1. If they already exist, delete the databases *WidgetSales* and *WidgetDeploy* from your SQL Server.
2. Click here to view the SQL creation script for the databases.
3. Copy the script, paste it in your SQL application, and then run it.
   The databases are created and populated with data.

Specifying the package contents

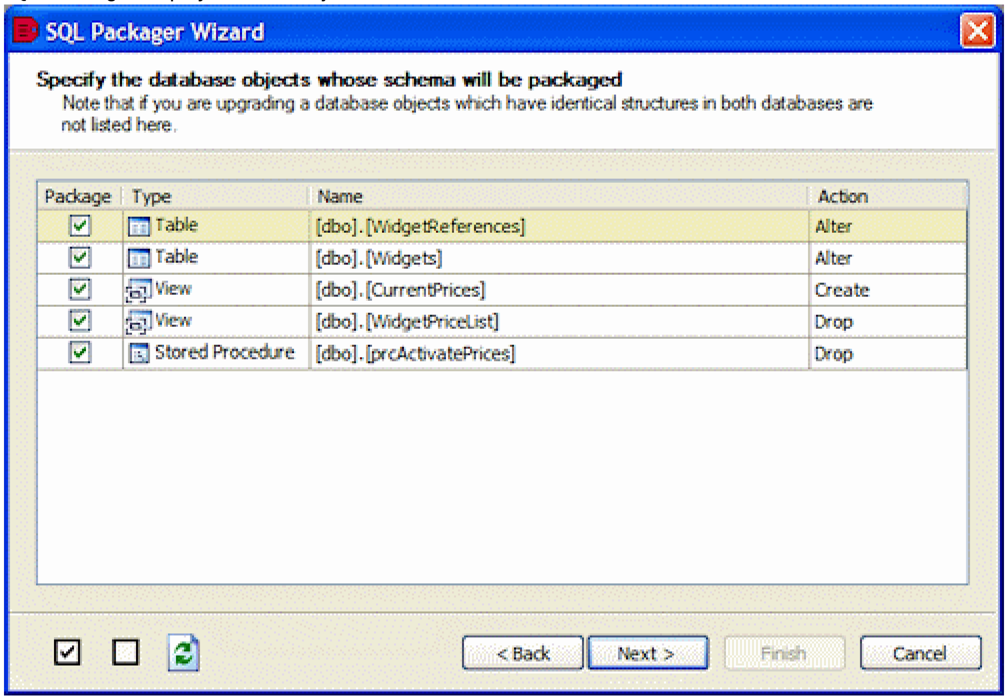
1. If you have not yet started SQL Packager, select it from your **Start** menu; if SQL Packager is already running, click
   New Project.
2. On the **Choose a project type** page of the Packager Wizard, select **Package an upgrade to a database**, and click **Next**.
   The **Choose databases to package into an upgrade** page of the Packager Wizard is displayed.
3. In the **Server** box, under **Latest Version Database**, type or select the name of the SQL Server on which you created the databases.
4. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
5. In the **Database** box, type or select *WidgetDeploy*.
   If *WidgetDeploy* is not displayed in the **Database** list, right-click in the **Database** box and click **Refresh** or scroll to the top of the list and click **Refresh**.
6. Under **Database to upgrade**, in the **Server** box, type or select the name of the SQL Server.
7. Select the authentication method, and for **SQL Server authentication** enter the **User name** and **Password**.
8. In the **Database** box, type or select *WidgetSales*.
   If *WidgetSales* is not displayed in the **Database** list, right-click in the **Database** box and click **Refresh** or scroll to the top of the list and click **Refresh**.


Choose databases to package into an upgrade
Specify locations and names of the two versions of the database that you would like to package into an upgrade.

Latest Version Database
Server: (local)
[✓ Windows authentication] SQL Server authentication
User name:
Password:
Database: WidgetDeploy

Database to upgrade
Server: (local)
[✓ Windows authentication] SQL Server authentication
User name:
Password:
Database: WidgetSales

9. Click **Next**.
SQL Packager displays a message dialog box while it analyzes the database structure.
If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that you choose the databases. For this example, leave the setting as it is.
10. Click **OK** to close the message box.
SQL Packager displays a list of objects whose structure differs in the databases.

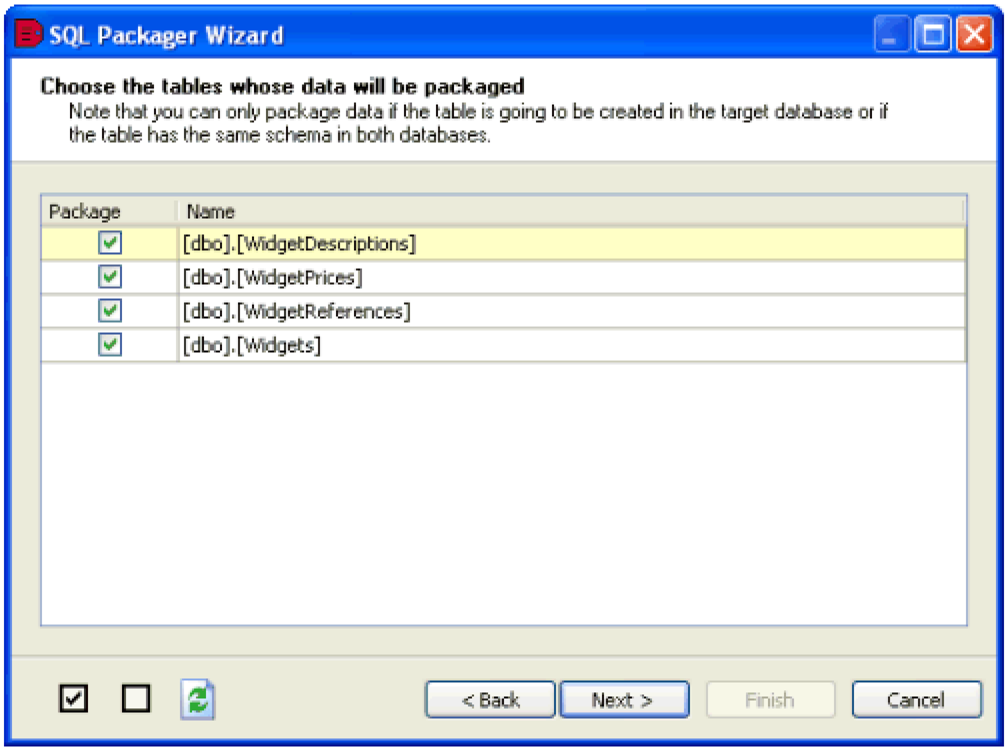

Specify the database objects whose schema will be packaged
Note that if you are upgrading a database objects which have identical structures in both databases are not listed here.

Package	Type	Name	Action
✓	Table	[dbo].[WidgetReferences]	Alter
✓	Table	[dbo].[Widgets]	Alter
✓	View	[dbo].[CurrentPrices]	Create
✓	View	[dbo].[WidgetPriceList]	Drop
✓	Stored Procedure	[dbo].[prcActivatePrices]	Drop

You can choose which objects to package. For more information, see Specifying the package contents.
For this example, leave all the check boxes selected so that all of the objects are updated when the package is run.
The **Action** column indicates the action that will be taken on *WidgetSales* to make it identical to *WidgetDeploy*.

> The upgrade not only creates new objects in *WidgetSales*; it also alters and drops objects from *WidgetSales* to make it identical to *WidgetDeploy*. For example, the *WidgetPriceList* view will be dropped from *WidgetSales*.

11. Click **Next**.
SQL Packager displays a list of the tables that contain data that can be packaged.



Choose the tables whose data will be packaged
Note that you can only package data if the table is going to be created in the target database or if the table has the same schema in both databases.

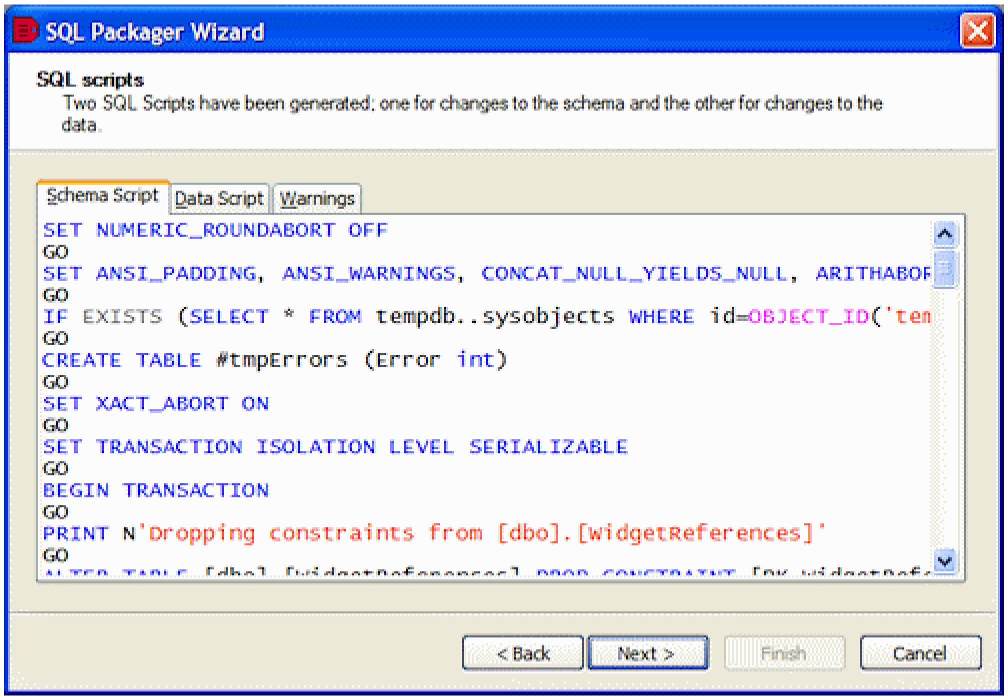
Package	Name
✓	[dbo].[WidgetDescriptions]
✓	[dbo].[WidgetPrices]
✓	[dbo].[WidgetReferences]
✓	[dbo].[Widgets]

You can choose the tables for which you want to package data. For more information, see Specifying the package contents.
For this example, leave all the check boxes selected so that all the data in the tables will be updated when the package is run.

12. Click **Next**.
    SQL Packager displays a message dialog box while it generates the SQL script.
    If you select the **Close message box on completion** check box, SQL Packager closes this message dialog box automatically the next time that it generates the SQL script. For this example, leave the setting as it is.
13. Click **OK** to close the message box.

Previewing the SQL Scripts

The **SQL scripts** page is displayed.



SQL scripts
Two SQL Scripts have been generated; one for changes to the schema and the other for changes to the data.

```
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABOR
GO
IF EXISTS (SELECT * FROM tempdb..sysobjects WHERE id=OBJECT_ID('tem
GO
CREATE TABLE #tmpErrors (Error int)
GO
SET XACT_ABORT ON
GO
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
GO
BEGIN TRANSACTION
GO
PRINT N'Dropping constraints from [dbo].[WidgetReferences]'
GO
```

The **SQL scripts** page displays the following tabs:

- **Schema Script** displays the SQL code to update the structure in *WidgetSales* so that it is identical to *WidgetDeploy*
- **Data Script** displays the SQL code to update the data in *WidgetSales* so that it is identical to *WidgetDeploy*
- **Warnings** provides details about unexpected behavior that may occur when you run the package

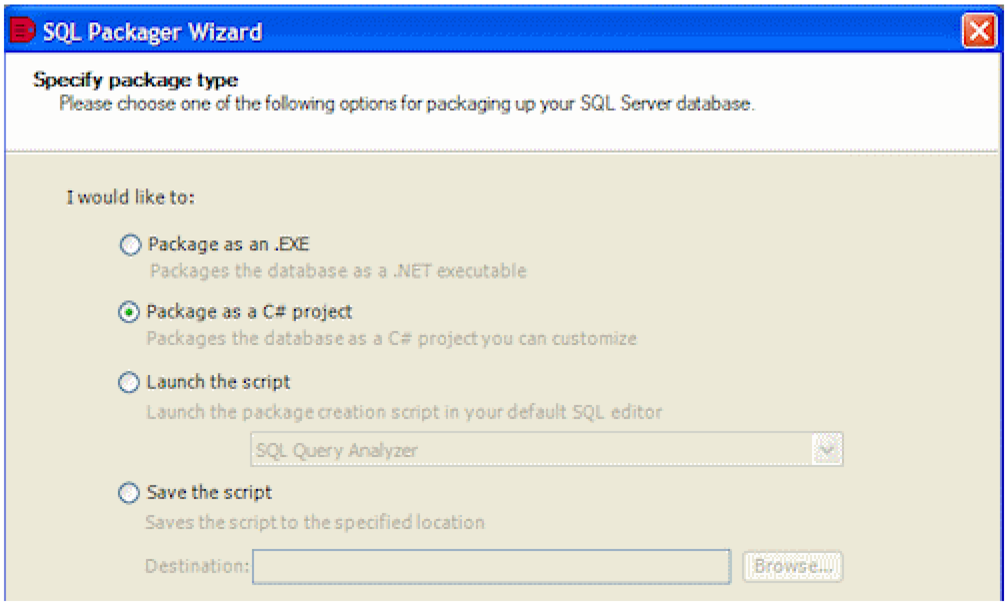
In this example, SQL Packager displays a warning to inform you that it cannot use the `ALTER TABLE` command to change the IDENTITY column, so the package will rebuild the WidgetReferences table. Warnings are displayed whenever tables require rebuilding as these may be slow operations.

If required, you can save the scripts from the next page of the Packager Wizard.

When you have finished reviewing the scripts, click **Next**.

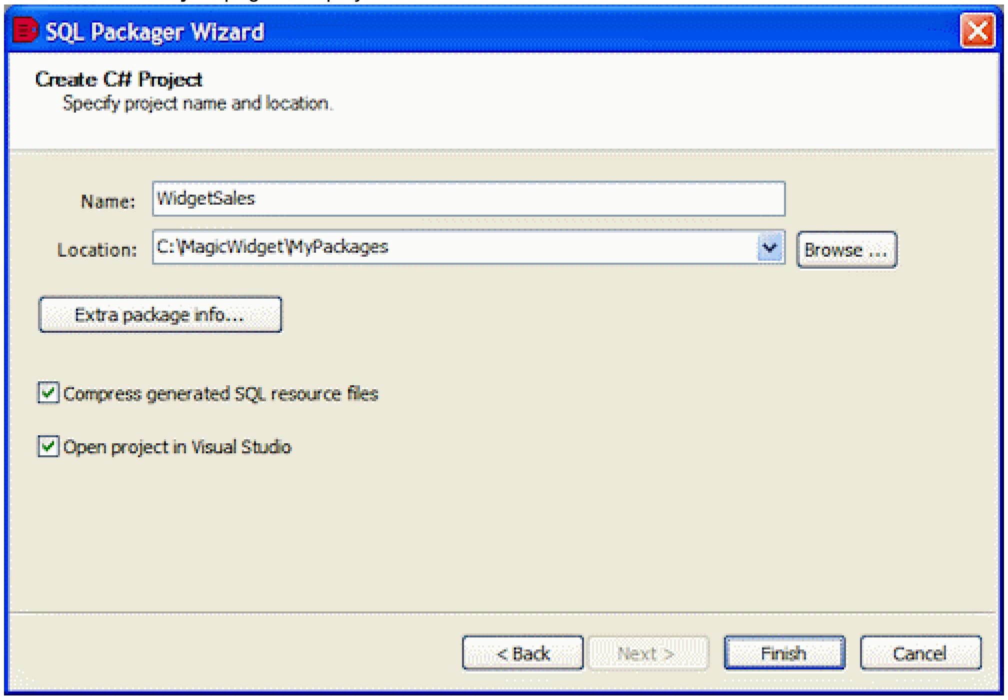
Generating the package

The **Specify package type** page is displayed.


SQL Packager Wizard
Specify package type
Please choose one of the following options for packaging up your SQL Server database.
I would like to:
Package as an .EXE
Packages the database as a .NET executable
Package as a C# project
Packages the database as a C# project you can customize
Launch the script
Launch the package creation script in your default SQL editor
SQL Query Analyzer
Save the script
Saves the script to the specified location
Destination:
Browse...

In this example, we will create a C# project. For an example of how to generate a .NET executable, see Packaging a database as an .EXE

1. Ensure that **Package as a C# project** is selected, and click **Next**.
   The **Create C# Project** page is displayed.



Name: WidgetSales
Location: C:\MagicWidget\MyPackages
[✓] Compress generated SQL resource files
[✓] Open project in Visual Studio

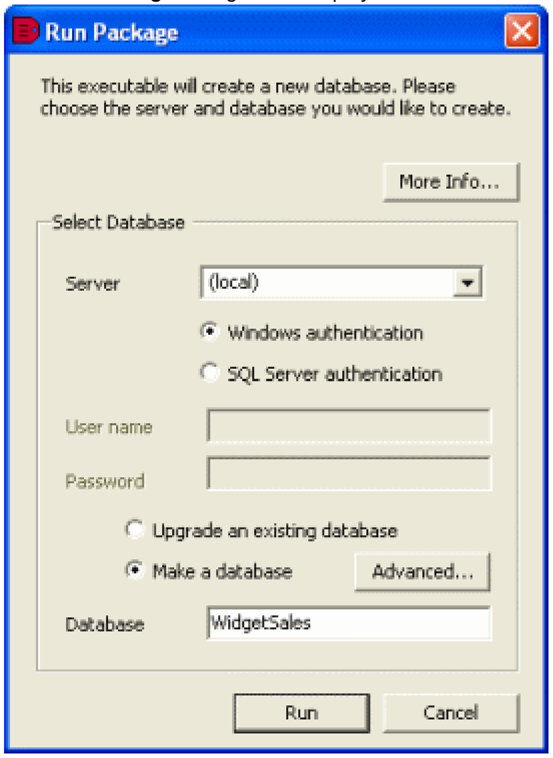
2. Enter a name and location for your package.
3. Leave the **Compress generated SQL resource files** check box selected so that your package will be compressed.
   The generated files will be compressed to approximately 75% of their original size.

   > Compressing the resource files means that you cannot edit the resource files or add resource files to the C# project.

4. Ensure the **Open project in Visual Studio** check box is selected, and click **Finish**.
   A message dialog box informs you that the project is created in the location you specified. Click **OK** to close it. Visual Studio .NET is launched with the project.

Running the package

To compile and run the project, in Visual Studio .NET 2005 or later:

1. Press **F5**, or on the **Debug** menu, click **Start**.
   The **Run Package** dialog box is displayed.
   
   Run Package

   This executable will create a new database. Please choose the server and database you would like to create.

   Select Database

   Server: (local)
   (● Windows authentication) ○ SQL Server authentication
   User name:
   Password:
   ○ Upgrade an existing database
   ● Make a database
   Database: WidgetSales

   You use this dialog box to specify details of the database that will be upgraded when the package is run.
2. Select the **Server** for *WidgetSales*, and if required, enter the authentication details.
3. Click **Run**.
   A message dialog box is displayed for you to confirm that you want to continue. Click **Yes**.
4. When WidgetSales is upgraded, a message dialog box confirms that the package has run successfully. Click **OK** to close it.

You can use your SQL application to check that the database has been changed as you expect. If you have purchased Red Gate **SQL Compare** you can compare the databases' structure to confirm that they are identical; if you have purchased **SQL Data Compare**, you can compare the data to confirm it is identical.

Getting more out of SQL Packager

- Upgrading the database structure and data
- Upgrading databases on different SQL Server versions
- SQL comparison and synchronization automation capabilities

Upgrading the database structure and data

When you use SQL Packager to upgrade the data in a database, you can select data only for those tables whose structure is identical.

If both the schema and the data has been updated for a particular table, and the schema changes include new columns that do not allow null values, you will have to run two packages; the first package to update the schema, and the second to update the data.

For example, the previous version of the database is called **DatabaseOld**, and the latest is called **DatabaseNew**. To upgrade **DatabaseOld**:

1. Create a package to upgrade only the schema of **DatabaseOld**:
   a. On the **Choose databases to package into an upgrade** page of the Packager Wizard, select **DatabaseOld** as the database to upgrade, and **DatabaseNew** as the latest version.
   b. On the **Specify the database objects whose schema will be packaged** page, select all the objects to package their schema.
   c. On the **Specify the tables whose data will be packaged** page, clear the selection for all of the tables so that no data is packaged.
   d. Generate the package (for example **Package1**).
2. Run **Package1** on **DatabaseOld**.**DatabaseOld** now contains the upgraded schema, but still has the old data.
3. Create a package to upgrade the data in **DatabaseOld** with the data in **DatabaseNew**:
   a. On the **Choose databases to package into an upgrade** page of the Packager Wizard, select **DatabaseOld** as the database to upgrade, and **DatabaseNew** as the latest version.
   b. On the **Specify the database objects whose schema will be packaged** page, no objects will be available for schema packaging because they are now identical.
   c. On the **Specify the tables whose data will be packaged** page, select all of the tables so that all data is packaged.
   d. Generate the package (**Package2**).
4. Run **Package2** on **DatabaseOld** to complete the upgrade.

If you need to upgrade a number of databases, you should deploy both packages and run **Package 1** followed by **Package 2**.

Upgrading databases on different SQL Server versions

This topic provides additional information for you if you are upgrading a database that is on a different version of Microsoft® SQL Server from the source database.

SQL Server 2008, SQL Server 2005, and SQL Server 2000

You can upgrade to a SQL Server 2008 database from a SQL Server 2005 or SQL Server 2000 database. Upgrading from SQL Server 2005 requires no additional actions. Upgrading from SQL Server 2000 may require changes to the packaging options.

For upgrading the database structure:

If you are upgrading to a SQL Server 2008 or SQL Server 2005 database from a SQL Server 2000 database, you must not change the default schema packaging options.

However, if your database is on a SQL Server with case-sensitive sort order, you must select **Treat items as case sensitive** in the schema packaging options.

If you are moving changes to a SQL Server 2000 database from a SQL Server 2005 database:

- SQL Packager may be unable to upgrade all objects. Warnings will be displayed where possible.
- SQL Packager cannot decrypt views, stored procedures, functions, DML triggers, and DDL triggers that are encrypted in a SQL Server 2008 or SQL Server 2005 database. Therefore, you cannot upgrade an object in a SQL Server 2000 database from an encrypted object in a SQL Server 2008 or SQL Server 2005 database.

When you create a default value or constraint in SQL Server 2008 or SQL Server 2005, the definitions of the default value or constraint are parsed and the parsed version is stored. The syntax of the SQL Server 2008 or SQL Server 2005 parsed version is not the same as the parsed version in SQL Server 2000. For example, in SQL Server 2005, **(1)** is parsed to **((1))**. If these are the only differences, SQL Packager considers the objects to be identical.

For upgrading the data:

- You can upgrade CLR data in a SQL Server 2008 or SQL Server 2005 database with values from a text or string data type in a SQL Server 2000 database. Ensure that the **Transport CLR data types as binary** data packaging option is not selected.
  SQL Packager considers the collation order for string data. Therefore, if the ordering is not the same as the CLR order, differences are reported.
- You can upgrade XML data in a SQL Server 2008 or SQL Server 2005 database with values from a text or string data type in a SQL Server 2000 database. SQL Packager will attempt to preserve white space. SQL Packager supports DTD (Document Type Definition), except for default attributes and entities.

  Some data, such as XML encoding and DTD, cannot be stored in the SQL Server 2008 or SQL Server 2005 representation. Therefore, if you convert data from a string data type to an XML data type, and then you convert back to a string data type, this information will be lost. SQL Packager considers the collation order for string data. Therefore, if the ordering is not the same as the XML order, differences are reported.

SQL Server 2008 and SQL Server 2005 compatibility level 80 databases

If a SQL Server 2008 or SQL Server 2005 database has its compatibility level set to 80, it conforms to strict rules for views, stored procedures, functions, and DML triggers. Therefore, upgrades may fail.

SQL comparison and synchronization automation capabilities

These command-line utilities are only available for licence-holders of the "Professional Edition" of the tools. Previous to version 5, these were licensed as part of "SQL Toolkit", which includes them as well as API access to the comparison and synchronization libraries.

In SQL Toolbelt Installer there are are three utilities - SQLCompare.exe; SQLDataCompare.exe and SQLPackager.exe. These are standalone executables capable of being placed into batch, VB Script, Perl or any other batch language, capable of performing all of the functions of the GUI driven equivalents.

The User Interface documentation and product help documentation do not describe all of the capabilities of these programs, but starting up an executable in a command prompt will reveal the all the usage syntax:

```
sqlcompare /? /verbose|more
sqldatacompare /? /verbose|more
sqlpackager /? /verbose|more
```

To produce a HTML document containing all the commands, use the `/html` switch:

```
sqlcompare /help /verbose /html >cmdhelp.html
```

Troubleshooting

This page provides information that may help you if you are having difficulties. You may also wish to refer to Understanding the results.

If you are asked to send Red Gate support a log file, see Logging and log files.

Common issues

- SQL Packager cannot connect to the SQL Server
- Missing tables in Packager Wizard
- Identity columns created or upgraded even though they are excluded
- Identical CLR data is flagged as different
- CLR data has not been upgraded
- SQL Packager sometimes creates DLL files with the executable
- Creating a database for an earlier version of SQL Server
- Rollback on script cancellation or failure
- Reseed applies "incorrect" identity values
- Primary keys, indexes, or unique constraints are not dropped
- Data has not been upgraded
- Package execution failing because the login already has an account under a different user name
- Package created from script always creates new database
- SQL Packager not keeping the READ_COMMITTED_SNAPSHOT SQL option ON
- Setting the database options for a New Database

Error messages

- "Value cannot be null" error creating data script
- Package executable cannot load zlib1.dll
- Invalid SQL when synchronizing an index to a scripts folder when data compression is specified
- Insufficient disk space
- A duplicate object name has been found
- User already exists in database
- DEFAULT_SCHEMA clause cannot be used with a Windows group

Common issues

- SQL Packager cannot connect to the SQL Server
- Missing tables in Packager Wizard
- Identity columns created or upgraded even though they are excluded
- Identical CLR data is flagged as different
- CLR data has not been upgraded
- SQL Packager sometimes creates DLL files with the executable
- Creating a database for an earlier version of SQL Server
- Rollback on script cancellation or failure
- Reseed applies "incorrect" identity values
- Primary keys, indexes, or unique constraints are not dropped
- Data has not been upgraded
- Package execution failing because the login already has an account under a different user name
- Package created from script always creates new database
- SQL Packager not keeping the READ_COMMITTED_SNAPSHOT SQL option ON
- Setting the database options for a New Database

SQL Packager cannot connect to the SQL Server

SQL Server doesn't exist or access is denied.

Possible causes

- SQL Server may be offline
- Incorrect port

How to fix

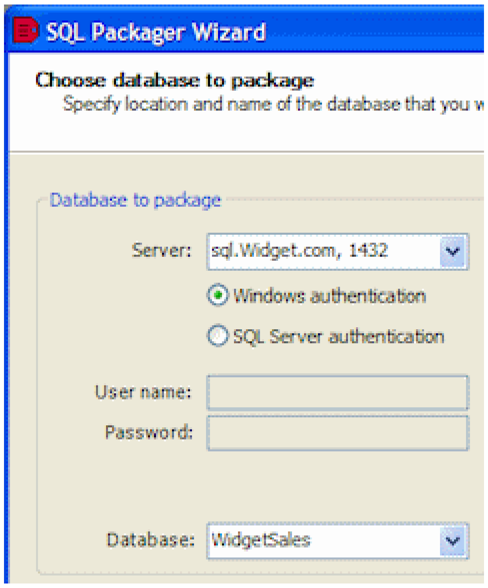
1. Check that the SQL Server is online and that the SQL Server name is listed in your LAN by *pinging* the address.
   For example, open a command prompt and run the following command:

   ```
   ping <ServerName>
   ```

   where *ServerName* is the name of your SQL Server.
2. If the SQL Server is online, check that you are connecting to the correct port.
   If your SQL Server is not running on the default port (1433), type the following in **Server**:

   ```
   <ServerName>,<Port>
   ```

   where *Port* is the number of the port on which your SQL Server is running

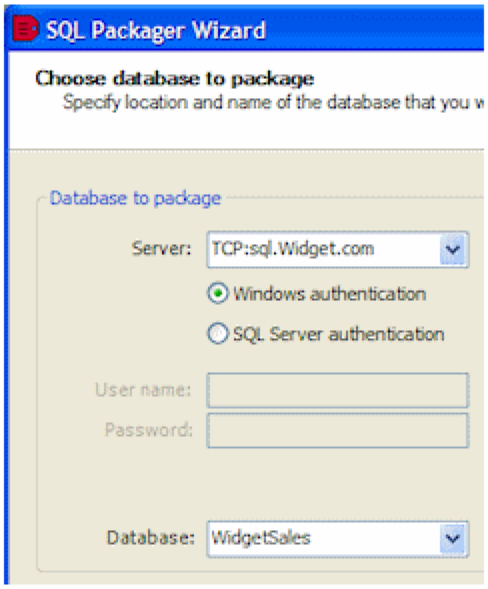
   
   SQL Packager Wizard
   Choose database to package
   Specify location and name of the database that you w
   Database to package
   Windows authentication
   SQL Server authentication
   User name:
   Password:

   Server: sql.Widget.com, 1432
   Database: WidgetSales

   If you are sure that you are connecting to the correct port, force SQL Packager to use the TCP network protocol when it makes the connection, by typing the following:

   ```
   TCP:<ServerName>
   ```

   For example:



SQL Packager Wizard

Choose database to package

Specify location and name of the database that you w

Database to package

Server: TCP:sql.Widget.com

(● Windows authentication) SQL Server authentication

User name:

Password:

Database: WidgetSales

If you continue to experience problems, please contact Red Gate Support.

Missing tables in Packager Wizard

If you are creating a package to upgrade a database, you can select tables to package their data only if they have:

- similar names
  You can set SQL Packager so that it ignores the case of object names, and spaces or underscores in object names by using the data packaging options.
- the same owner (case-sensitive)
- similar schema
- a matching primary key, unique index, or unique constraint that has the same name in both databases

If the tables are very different, for example if primary keys or column data types are different, SQL Packager cannot upgrade the data. Red Gate offers **SQL Compare**, which will synchronize the schema of two databases. You can then use SQL Packager to package an upgrade to the data.

Alternatively, you could use SQL Packager to upgrade the schema, then generate a second package to upgrade the data. For details, see Upgrading the database structure and data.

In addition, if the structure of the databases that you have selected has changed while you are working on a project, those changes are not automatically shown in the **Choose the tables whose data will be packaged** page of the Packager Wizard. To refresh the page, click

; you can then select the tables. For more information, see Specifying the package contents.

Identity columns created or upgraded even though they are excluded

If you clear the data packaging option **Include identity columns**, but an identity column is the key used to match records, the identity column is included in the package.

Identical CLR data is flagged as different

SQL Packager considers the collation order for CLR and XML data. Therefore, if the ordering is not identical, differences are reported.

CLR data has not been upgraded

Data for CLR types can be stored as string or binary values. When CLR data is compared, SQL Packager always compares the binary representations. However, by default, when CLR data is upgraded, SQL Packager updates the string representations, because binary formats are not always compatible.

How to fix

If you know that the binary formats are compatible, you can select the **Transport CLR data types as binary** data packaging option to force SQL Packager to update the binary representations.

SQL Packager sometimes creates DLL files with the executable

SQL Packager creates a dynamic-link library (DLL) file in addition to the executable when the package exceeds 100MB.

For each additional 100MB, another DLL file is created. You must distribute the executable and all the DLL files to run the package.

Creating a database for an earlier version of SQL Server

SQL Packager creates databases in a way that is compatible with the database being packaged. For example, if the database resides on the SQL 2000 platform, the database scripts will be created in the SQL 2000 syntax. If Packager is configured to create an "upgrade" package, then it can change the T-SQL syntax so that it conforms to the target server platform.

To create a database package using the SQL syntax different from the version of the source database, it is necessary to have an installation of the version of SQL Server that you would like to deploy to.

1. On the desired SQL Server, create a new database and leave it empty.
2. Launch SQL Packager and choose your source database on the left-hand side.
3. Choose the option to create an "upgrade to an existing database" and specify the empty database that you had just created on the right.

SQL Packager will produce a script to create all objects in dependency order, and use syntax that is compatible with the target server platform.

Where possible, new features such as CLR assemblies are filtered out because they are incompatible with SQL Server 2000. Sometimes this is not possible, for instance if a stored procedure relies on a CLR function. Since the function cannot be created, the stored procedure cannot be successfully scripted. Analyzing your databases for these conditions first is recommended.

If you continue to experience problems, please contact Red Gate Support.

Rollback on script cancellation or failure

If a script fails, or if it is cancelled, in most circumstances changes are rolled back. SQL Packager uses *transactions* to do this. However, there are some circumstances in which this is not possible:

- if full-text information must be altered
  For example, within a transaction, catalogs cannot be dropped, and indexing cannot be added to a column.
- if users and roles need to be created, altered, or deleted
  For example, within a transaction, a user cannot be created, or added to a role.

In these cases, SQL Packager rolls back all the changes that it can. Your database will be in an undetermined state.

If the data packaging script fails, only the data changes are rolled back; the changes made by the schema packaging script are not rolled back.

If you have selected the schema packaging option **Do not include plumbing for transactional synchronization scripts** to remove transactions from the schema packaging script, or cleared the data packaging option **Use transactions in SQL scripts** to remove transactions from the data packaging script, no changes are rolled back when the script is cancelled or fails. This may be useful if you want to run a script up to the point of failure, for example for debugging.

SQL Packager always warns you if it is unable to roll back changes.

Reseed applies "incorrect" identity values

After synchronizing data using SQL Data Compare or SQL Packager, the next time data is inserted into one of the synchronized tables, an error may result claiming that the automatically-generated value violates a primary key or unique constraint.

Cause

SQL Data Compare will "reseed" identity values automatically after synchronization if the **Reseed Identity columns** option in the Options panel of the project is selected. It does this by gathering the current identity value from the "source" table and applying it to the "target" table. In this way, any identity columns with a primary key constraint will continue generating valid identity values.

There are a few circumstances, however, where the current identity value in the target database will become or remain invalid:

- A data update was taking place before the synchronization or at the same time as the data comparison.
- The direction of synchronization was changed.
  in that case the databases need to be recompared because the identity value is gathered at registration-time.
- There are no differences in the table, at that point SDC does not reseed and the identity values remain different.
- All records were deleted from the source table using a `DELETE` query.
  In this case the identity is not rolled backwards and SQL Data Compare will synchronize the current identity, which may be higher than expected. Using `TRUNCATE TABLE` will roll the identity back to the base seed value.

If you continue to experience problems, please contact Red Gate Support.

Primary keys, indexes, or unique constraints are not dropped

If you select the data packaging option **Drop primary keys, indexes, and unique constraints**, note that primary keys, indexes or unique constraints that are selected as comparison keys are not dropped.

Data has not been upgraded

This may occur if:

- there are triggers defined on the tables
  If you have a trigger defined on a table that inserts data into another table on `INSERT, DELETE,` or `UPDATE,` the data in the tables will change as the upgrade is run, which will cause unpredictable results.
  To avoid this, select the **Disable DML triggers** data packaging option before you generate the package.
- primary keys are defined on columns with differing collation order
  If you upgrade tables that have primary keys defined on columns that have different collation order, SQL Packager may produce unpredictable results.
- columns contain timestamp data
  SQL Packager can not upgrade data in timestamp columns.
- tables do not have identical schema
  If the structure of the tables you are upgrading is not identical, SQL Packager may produce unpredictable results.

If you continue to experience problems, please contact Red Gate Support.

Package execution failing because the login already has an account under a different user name

When executing a SQL Packager executable, the program can fail with the error "The login already has an account under a different user name."

Cause

This happens because SQL Packager includes dependencies, and if there is a difference in a database role or a new database role is being created, it will attempt to create the user as part of the synchronization. This is why you may continue getting this error even after recreating the package with the offending user name excluded.

How to fix

To solve this problem, deselect the option **Include dependencies** in SQL Packager's options and build a new package executable. You will first need to upgrade to the latest version, SQL Packager 6.4, if you are not currently using it.

After upgrading and removing the dependencies scripting from the options and excluding all users from the package that already exist in the database, the error should not occur.

If you continue to experience problems, please contact Red Gate Support.

Package created from script always creates new database

Note: This problem is fixed in version 6.2

When running a SQL Package executable from the command line, a new database is always created regardless of whether or not the `/makedatabase` switch is specified. This happens only when the package had been created by packaging an existing SQL script file.

Cause

There is a bug in the SQL Packager code template that will cause the package to create a database whenever the saved package properties specify a new database package, and the command-line arguments do not override this. Packaging a script always behaves as if a new database will be created.

How to fix

As a workaround, the "SQL Packager Code Templates" files can be modified so that the presence command-line /makedatabase switch will override the database creation setting saved into the package. To do this:

1. Use Windows Explorer to browse to *%programfiles%\red gate\sql packager 6\SQL Packager Code templates\C#*
2. Open *PackageExecutor.cs* using Notepad
3. From the **Edit** menu, choose **Go To** and go to line 421
4. Open a new line above the line beginning with "m_MakeDatabase = "
5. Insert the following text in the new line:

```
if (!PackageUtils.ConsoleMode)
```

All SQL Packager projects created after this modification will that are run from the command prompt will only create a new database when the `/makedatabase` switch is used.

SQL Packager not keeping the READ_COMMITTED_SNAPSHOT SQL option ON

Unfortunately it is not possible to configure SQL Packager to create a database setting `READ_COMMITTED_SNAPSHOT`. When you package a database for a new install, the transaction isolation level will always be unspecified, which will more than likely set it to "read committed".

How to fix

There are a couple of workarounds:

1. Use a post SQL Script:
   If you run the package from the command line, you can specify `/postsql:<filename>`. You could use a sql file that contains the `ALTER DATABASE [dbname] SET READ_COMMITTED_SNAPSHOT ON`, which will be run after the database is created.
2. Edit the SQL Package and add the code to set the user options.
   To do this, output the package as a C# project, then in the PackageExecutor class add the following code after the dboptions were set:

```
sqlCommand.CommandText = "Alter database " + m_DatabaseName + " SET
ALLOW_SNAPSHOT_ISOLATION ON";
sqlCommand.ExecuteNonQuery();
sqlCommand.CommandText = "Alter database " + m_DatabaseName + " SET
READ_COMMITTED_SNAPSHOT ON";
sqlCommand.ExecuteNonQuery();
```

This sets the isolation level to 'read committed snapshot' confirmed by checking `DBCC USEROPTIONS` after the database is installed..

Setting the database options for a New Database

When using SQL Packager to package a new database, you will notice that the databases are created with a default options set, rather than the same options as the source database.

The default options used by SQL Packager for a new database are:

- auto create statistics
- auto update statistics

If you want to set options other than the default options, you can add them to `PACKAGE_NEW_DBOPTIONS` in *PackageProperties.resx* when you export the package as a C# project, using the built-in Visual Studio resource editor.

The options will need to be semi-colon separated, and only options that can be set through sp_dboption are valid. See http://msdn.microsoft.com/en-us/library/aa933268(v=sql.80).aspx for more details.

Error messages

- "Value cannot be null" error creating data script
- Package executable cannot load zlib1.dll
- Invalid SQL when synchronizing an index to a scripts folder when data compression is specified
- Insufficient disk space
- A duplicate object name has been found
- User already exists in database
- DEFAULT_SCHEMA clause cannot be used with a Windows group

"Value cannot be null" error creating data script

When creating a new database package, SQL Packager may return the following error during the creation of the data script:

"Value cannot be null. Parameter name: format"

Cause

SQL Packager attempts to create a script to populate a new table, but if the table contains a column with a Binary Large Object (BLOB) datatype such as image or text, the scripting process fails because it has no index to use as the basis for the query to insert the BLOB data. If a unique index or primary key is created for the table, the scripting process will succeed.

How to fix

To work around this issue, therefore, you could create a primary key or unique index on the table, or exclude the table from the list of tables whose data you want to script.

If more than one of these columns exists, you may identify it using the following (SQL Server 2005 and up) query:

```
SELECT NAME, object_id FROM sys.objects WHERE TYPE='U' AND OBJECT_ID IN (
SELECT object_id FROM sys.indexes WHERE TYPE = 0
AND OBJECT_ID NOT IN (SELECT object_id FROM sys.indexes WHERE TYPE <> 0 OR is_unique=
1))
and OBJECT_ID in (select OBJECT_ID from sys.columns where system_type_id in (35, 165,
99, 34, 173))
```

If you continue to experience problems, please contact Red Gate Support.

Package executable cannot load zlib1.dll

When running a package executable created using the SQL Packager API using compression on a 64-bit version of Windows, the package may throw the following error when executed on another computer hosting a 64-bit version of Windows:

Unable to load DLL 'zlib1.dll': The specified module could not be found. (Exception from HRESULT: 0x8007007E).

Cause / Possible causes

The problem is caused by the default 'Any CPU' configuration when the project is compiled on .NET. On a 64-bit system, this code will run as a 64-bit application. When the decompression library is linked to the package application, this is a 32-bit library, which cannot be loaded into an application that is 64-bit aligned.

How to fix

One possible solution is to output the package as a C# project rather than an executable, open it in Visual Studio 2003, open the project's properties, and change the project properties from 'Any CPU' to 'x86'. Then build the project. This will create a 32-bit version of the executable that can load the 32-bit version of Zlib1.dll.

Another possibility is to use Microsoft's Corflags.exe utility to set the package assembly's 32-bit flag in the assembly header as described here: http://msdn2.microsoft.com/en-us/library/ms164699(VS.80).aspx

SQL Packager version 6 and higher has renamed Zlib1.dll to RedGate.Compression.Zlib.dll, and this dll is set to JIT compile on 'Any CPU', so this problem will not occur when running SQL Package executables on 64-bit systems, regardless of whether they are built on a 64 or 32-bit system.

If you continue to experience problems, please contact Red Gate Support.

Invalid SQL when synchronizing an index to a scripts folder when data compression is specified

When attempting to synchronize a database to a scripts folder in SQL Compare, some existing SQL code in the scripts folder may cause SQL Compare to show a warning regarding invalid SQL. If this happens because an index specifies SQL Server 2008 data compression, this is a known issue in SQL Compare. For example, the following syntax will be flagged as invalid:

```
ALTER TABLE [dbo].[MyTable] ADD CONSTRAINT [PK_MyTable] PRIMARY KEY CLUSTERED
([MyColumn]) WITH (DATA_COMPRESSION = PAGE)
```

The warning message can be safely ignored, however the table will always be flagged as "different", even after synchronization because the invalid syntax warning will always cause SQL Compare to err on the side of caution and flag the object as different. We are aware of this and our engineers are going to fix it in a future version.

If you continue to experience problems, please contact Red Gate Support.

Insufficient disk space

SQL Packager may be unable to create or upgrade databases if there is insufficient disk space.

Cause

SQL Packager uses temporary files when it analyzes the databases to create the package. To successfully create a package, these temporary files require available disk space at least four times the size of the database you are packaging.

How to fix

You can decrease the size of the temporary files by selecting the **Use checksum comparison** data packaging option.

The location of the temporary files is defined by the **RGTEMP** environment variable, or the **TMP** variable if **RGTEMP** does not exist (see your Windows® documentation for information about environment variables).

> Changing the **TMP** variable will affect other programs that use the variable.

If you continue to experience problems, please contact Red Gate Support.

A duplicate object name has been found

SQL Packager displays this message when you connect to a database on a SQL Server that uses case-sensitive sort order and you have not selected the **Treat items as case sensitive** schema packaging option.

How to fix

Select the case sensitive schema packaging option.

If you continue to experience problems, please contact Red Gate Support.

User already exists in database

In Microsoft Windows, users are a composite of the domain name or computer name, and the user name, for example *Computer1\WindowsUser1* . SQL Packager references both parts of the name.

However, SQL Server references only the user name, so that *Computer1\User1* and *Computer2\User1* would be considered as the same. Therefore, if SQL Packager attempts to create a user for which the computer name is different but the user name is the same, SQL Server returns an error.

How to fix

To avoid this error, ensure that the user names are different for users that you want to upgrade.

If you continue to experience problems, please contact Red Gate Support.

DEFAULT_SCHEMA clause cannot be used with a Windows group

When running a schema synchronization script, the following error may occur, and the synchronization will fail.

The `DEFAULT_SCHEMA` clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys.

Cause

The cause of this issue is that SQL Compare and SQL Packager cannot reliably determine whether a Windows user is an actual user or a group. Windows groups can be mapped to SQL Server users just as Windows users, but a default schema cannot be specified for a Windows group. Unfortunately, not specifying a default schema for a Windows user can lead to other problems.

How to fix

The only possible workaround for this issue is to note the name of the group in the error dialog, and remove that SQL Server user from the synchronization.

Logging and log files

Log files collect information about the application while you are using it. These files are useful to us if you have encountered a problem.

By default, logging is disabled and no log files are stored.

Enabling logging

To enable logging, right-click the application title bar, click **Minimum Log Levels**, and then select the required log level:

? Unknown Attachment

Select a minimum log level based on how much information you need to be reported:

Verbose	Reports all messages in the log file.
Warning	Reports warning and error messages. For example, a warning message might report a handled exception, or a problem which does not prevent you from using the application.
Error	Reports serious and fatal errors. For example, an error message might report a failed operation.
No Logging	Disables logging.

Note that the selected log level may affect performance. Verbose logging reports all messages, and so writes the most information to disk and produces the largest log files.

If a problem has been resolved and you no longer require logging, you are recommended to select **No Logging**.

Locating the log files

To open the folder where the log files are stored, click **Locate Log Files**. By default the log files are located in:

%ALLUSERSPROFILE%\Application Data\Red Gate\Logs

To view the current log file in your default text editor, click **Open Current Log File**.

If the minimum log level is set to **No Logging**, you cannot locate or open log files from within the application.

Release notes and other versions

Version 8.0 (current)	December 3rd, 2014	Release notes	Documentation
Version 7.0	July 17th, 2013	Release notes	Documentation
Version 6.4	January 26th, 2011	Release notes	Documentation
Version 6.0	September 22nd, 2008	Release notes	
Version 5.5	July 28th, 2008	Release notes	Documentation (PDF)
Version 5.4	June 25th, 2007	Release notes	
Version 5.3	December 18th, 2006	Release notes	
Version 5.2	June 29th, 2006	Release notes	
Version 5.1	May 15th, 2006	Release notes	

If you need to install an old version of SQL Packager, go to Download old versions of products.

SQL Packager 6.4 release notes

January 26th, 2011

This is a maintenance release that fixes a number of issues in SQL Packager 6.0.

New features

- **Include Dependencies** option
- Uses the SQL Compare 8.50 and SQL Data Compare 8.50 engines.

Fixes

- SPA-566 IncludeData switch broken
- SPA-535 ERR Creating DB: Data is Null. This method or property cannot be called on Null values.
- SPA-548 Batch separation happens on GOTO commands in stored procedures
- SPA-549 Exception packaging with non-default options
- SPA-541 SQL Packager command line needs a substantial rewrite to incorporate SC and SDC 8 engines
- SPA-533 Add option to decrypt encrypted objects in SQL Server 2005 and 2008 databases.
- SPA-527 Validate that a database exists before running /presql script
- SPA-527 Removed the option for including comment headers in scripts
- SPA-534 Missing Collations running package
- SPA-536 Ignore STATISTICS_NORECOMPUTE property on indexes
- SPA-537 Split transactions into specified number of Mb
- SPA-538 Change the date format to cater for different collations and/or locale settings
- SPA-552 SQL Packager 6 command line asks for SQL Compare license
- SPA-555 CL using wrong credentials to register data comparison
- SPA-553 NullReferenceException packaging stored procedure script
- SPA-558 'Object reference not set to an instance of an object' when generating script
- SPA-509 Unable to package large batch
- SPA-554 SQL Packager Creating whitespace differences in packaged objects
- SPA-571 'The variable name '@pv' has already been declared'

SQL Packager 6.0 release notes

September 22nd, 2008

New features

This release enables two much requested features:

- packaging of externally generated SQL scripts into exe’s or C# projects
- the ability to edit the SQL script generated by SQL Packager before it is deployed

SQL Packager 5.5 release notes

July 28th, 2008

This is a minor upgrade.

Support for SQL Server 2008:

SQL Packager 5.5 is a minor release of SQL Packager, giving full support for SQL Server 2008. This support is based on the currently available release candidate. This version also adds better licensing handling, check for updates and installation experience.

SQL Packager 5.4 release notes

June 25th, 2007

Enhancements:

- Moved to web-based help

SQL Packager 5.3 release notes

December 18th, 2006

Enhancements:

- Support for Windows Vista
- Installation can now run quietly

Bug fixes:

- Improved support of non-compliant assemblies
- Improved .NET2 Snapshots with XML Schema Collections
- Improved Dependency calculating in Synchronization
- Support for multiple column INCLUDEs in INDEX es
- Improved support for Scripts created as Unicode when non-Latin collation
- Improved support for Foreign keys (now recreated with SET NULL or SET DEFAULT)

SQL Packager 5.2 release notes

June 29th, 2006

Bug fixes:

- Updated help
- Fixed problem when reading case-sensitive computed columns
- Fixed scripting issue when attempting to revoke permissions on newly built function
- Fixed "Specified Cast is not valid" exception when users do not have full permissions
- Fixed problem where occasionally an "index outside the bounds of the array" exception was thrown when reading CLR objects
- Fixed problem where if the “ignore constraint and index names” option was enabled objects could be missing from the synchronisation
- Fixed problem where if a constraint’s object text was greater than 4000 characters a duplicate object would be detected.
- TYPE COLUMN options now work correctly with varbinary(max)
- Calls to sp_addrolemember are now unicode compliant
- Context sensitive help nolonger throws an unhandled exception when running under .NET2
- Fixed problem where SQL Compare attempted to generate XML indexes on SQL2000 databases
- Fixed problem with UDTs and Indexed views
- No longer attempts to do invalid casts of alias types
- SQL Packager no longer includes data when the table schema is deselected
- Indexed Views are now displayed correctly
- Fixed problem when loading an old project, deselected tables were reselected
- It is now possible to disable the "Include Indexed views" option
- Added basic support for pre & post SQL scripts

SQL Packager 5.1 release notes

May 15th, 2006

Bug fixes:

- Problems reported since the original release