

1. SQL Compare 10 documentation	3
1.1 Requirements	4
1.2 New in SQL Compare 10	5
1.3 About SQL Compare	11
1.3.1 Third-party software used by SQL Compare	12
1.4 Installing	13
1.5 Licensing	14
1.5.1 Activating	15
1.5.2 Deactivating	21
1.5.3 Troubleshooting licensing and activation	24
1.5.4 Changes to distribution of command line	27
1.6 Upgrading	28
1.6.1 Using Check for Updates	29
1.6.2 Troubleshooting Check for Updates errors	31
1.7 Setting up the comparison	32
1.7.1 Working with projects	33
1.7.2 Setting data sources	35
1.7.3 Creating a new database	40
1.7.4 Mapping tables and columns	41
1.7.5 Mapping owners	43
1.7.6 Setting project options	44
1.7.7 Comparing databases on unconnected SQL Servers	51
1.7.8 Permissions required to use SQL Compare	52
1.8 Reviewing the comparison results	53
1.8.1 Viewing the comparison results	54
1.8.2 Viewing the SQL differences	61
1.8.3 Using filters	65
1.8.4 Comparison warnings	69
1.8.5 Understanding the comparison results	70
1.8.6 Generating a report	72
1.9 Deploying data sources	74
1.9.1 Backing up before deployment	76
1.9.2 Deployment warnings	78
1.9.3 Understanding the deployment	80
1.9.4 Using the Deployment Wizard	83
1.9.5 SQL Compare may drop and create CLR assemblies	90
1.9.6 Rollback on script failure or cancellation	91
1.10 Working with other data sources	92
1.10.1 Working with backups	93
1.10.2 Working with snapshots	94
1.10.3 Working with scripts folders	96
1.10.4 Using a DACPAC as a data source	99
1.11 SQL Server Management Studio add-in	101
1.11.1 Getting started with the add-in	102
1.11.2 Using SQL Compare projects	105
1.12 Using the command line	106
1.12.1 Command line basics	107
1.12.2 Integrating the command line with applications	110
1.12.3 Simple examples using the command line	111
1.12.4 Using XML to specify command line arguments	114
1.12.5 Example - selecting single tables for comparison	118
1.12.6 Example - selecting tables with unrelated names	120
1.12.7 Switches used in the command line	122
1.12.8 Options used in the command line	138
1.12.9 Exit codes used in the command line	146
1.13 Getting more from SQL Compare	149
1.13.1 Comparing databases on different SQL Server versions	150
1.13.2 Deploying to SQL Azure	151
1.13.3 Creating a rollback script	154
1.13.4 Logging and log files	155
1.13.5 Forcing SQL Compare and SQL Data Compare to use an encrypted connection	156
1.13.6 Copying the structure of a database	157
1.14 Worked examples	158
1.14.1 Worked example - comparing and deploying two databases	159
1.14.2 Worked example - using a scripts folder as a data source	165
1.15 Troubleshooting	172
1.15.1 Common issues	173
1.15.1.1 SQL Compare shows identical objects as different	174
1.15.1.2 Comparison seems slow	175
1.15.1.3 Errors in scripts folders	176
1.15.1.4 Can't compare encrypted text in SQL Server 2005 or 2008	177
1.15.1.5 Deploying a SQL 2000 compatible database from a SQL 2005 database using SQL Compare	178

1.15.1.6	When does the deployment process rebuild tables?	179
1.15.2	Error messages	180
1.15.2.1	A duplicate object name has been found	181
1.15.2.2	HTML reports are generated for identical comparisons	182
1.15.2.3	Could not enlist in a distributed transaction	183
1.15.2.4	Could not start a transaction for OLE DB provider	184
1.15.2.5	Error Comparing DB1 vs DB2. Cannot insert the value NULL into column 'YourColumn', table 'database.dbo.tmp_rg_xx_MyTable'; column does not allow nulls. INSERT fails	185
1.15.2.6	SQL Server doesn't exist or access is denied	186
1.15.2.7	The DEFAULT_SCHEMA clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys	188
1.15.2.8	The operation could not be performed because the OLE DB provider 'SQLOLEDB' was unable to begin a distributed transaction...	189
1.15.2.9	This SQL Server has been optimized for X concurrent queries. This limit has been exceeded by X queries and performance may be adversely affected	190
1.15.2.10	User or role already exists in the current database / The login already has an account under a different user name	191
1.16	Release notes and other versions	192
1.16.1	SQL Compare 10.7 release notes	193
1.16.2	SQL Compare 10.4 release notes	195
1.16.3	SQL Compare 10.3 release notes	196
1.16.4	SQL Compare 10.2 release notes	197
1.16.5	SQL Compare 10.0 release notes	199
1.16.6	SQL Compare 9.0 release notes	200
1.16.7	SQL Compare 8 release notes	201
1.16.8	SQL Compare 7 release notes	203
1.16.9	SQL Compare 6 release notes	204
1.17	* Work in progress pages *	205
1.17.1	SQL Compare 11 beta release notes	206

# SQL Compare 10 documentation

## About SQL Compare

With SQL Compare, you can compare and deploy the structure of two Microsoft SQL Server databases.

SQL Compare includes a free add-in for SQL Server Management Studio you can use to compare and deploy databases from SSMS. For more information, see [Getting started with the add-in](#).

## Quick links

[SQL Compare product page](#)

**Latest version:** [Release notes for SQL Compare 10.7](#)

[Deploying the data sources](#)

[Using the Deployment Wizard](#)

[Using the command line](#)

# Requirements

To use SQL Compare you need:

- One of the following Microsoft Windows operating systems:
  - Windows Server 2003
  - Windows 7
  - Windows Server 2008
  - Windows 8
- [Microsoft .NET Framework 3.5](#) or later
- 256 MB RAM
- 200 MB hard disk space

The following versions of Microsoft SQL Server are supported:

- SQL Server 2005
- SQL Server 2008
- SQL Server 2008 R2
- SQL Server 2012
- Windows Azure SQL Database
- SQL Server on Amazon RDS

# New in SQL Compare 10

- Migration script support
- Table mapping
- Creating a new database to compare
- Comparing revisions from SQL Source Control
- New command line switches
- New project options
- Connecting to SQL Server 2012
- Transaction Isolation Level application option

## Migration script support

With SQL Source Control 3.0, you can create migration scripts between two versions of a database. Migration scripts are customizable change scripts that are committed to source control and re-used in deployment.

This can be useful, for example, if you add a NOT NULL constraint to a column in development and want to deploy this change to your testing environment. If you try to deploy this change using SQL Compare, the deployment script will fail because a default value is required; you can specify this value using a migration script.

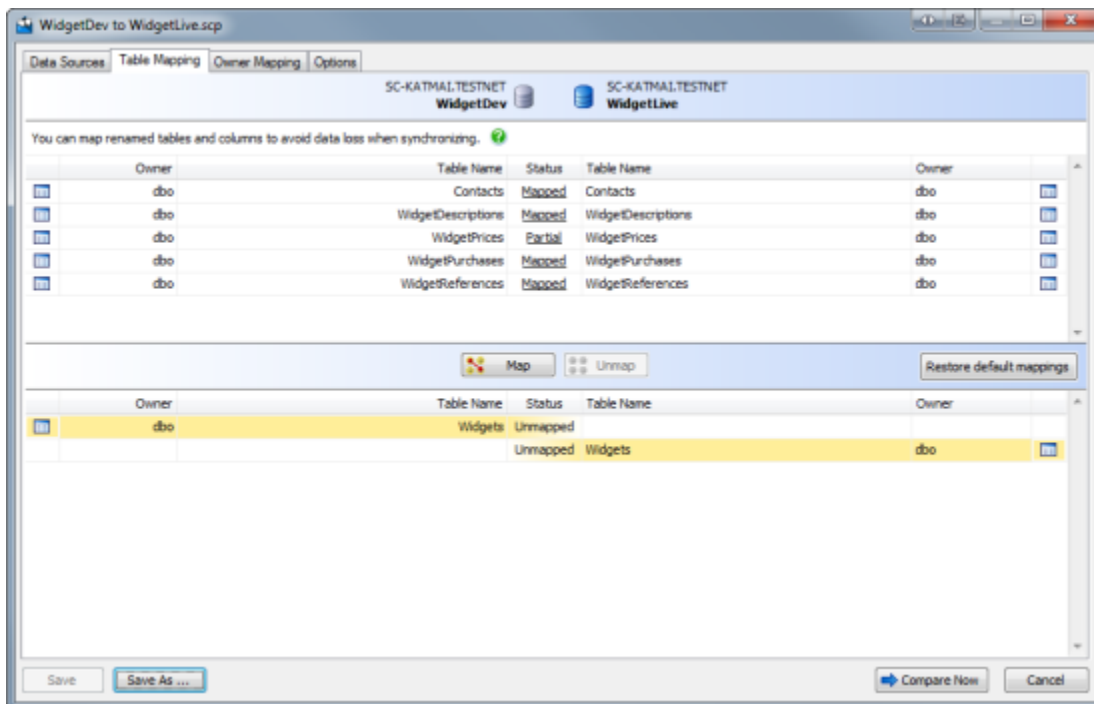
For more details, see: [Working with migration scripts](#).

## Table mapping

With SQL Compare 10, you can map together tables and columns with different names. This can be useful to prevent data loss when deploying tables or columns that have been renamed.

For example, if you map the table *TableA* in the source to the same table that has been renamed as *TableB* in the target, SQL Compare will compare the table as an object that exists in both data sources. When you deploy the table, the name change will be scripted using the *sp\_renam*e system stored procedure; the table isn't dropped and re-created.

To compare tables and columns that aren't automatically mapped, click the **Table Mapping** tab of the Project Configuration dialog box:



The upper pane displays tables that are fully **Mapped** or have **Partial** mapping. The lower pane displays Unmapped tables.

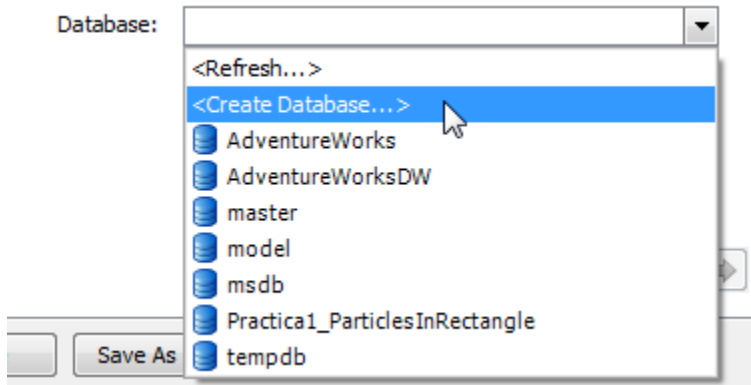
- If a table has a Partial mapping, some of its columns aren't mapped, and can't be compared.
- To set the column mappings for a table, click the **Status** link for the object you want to re-map.

## Creating a new database to compare

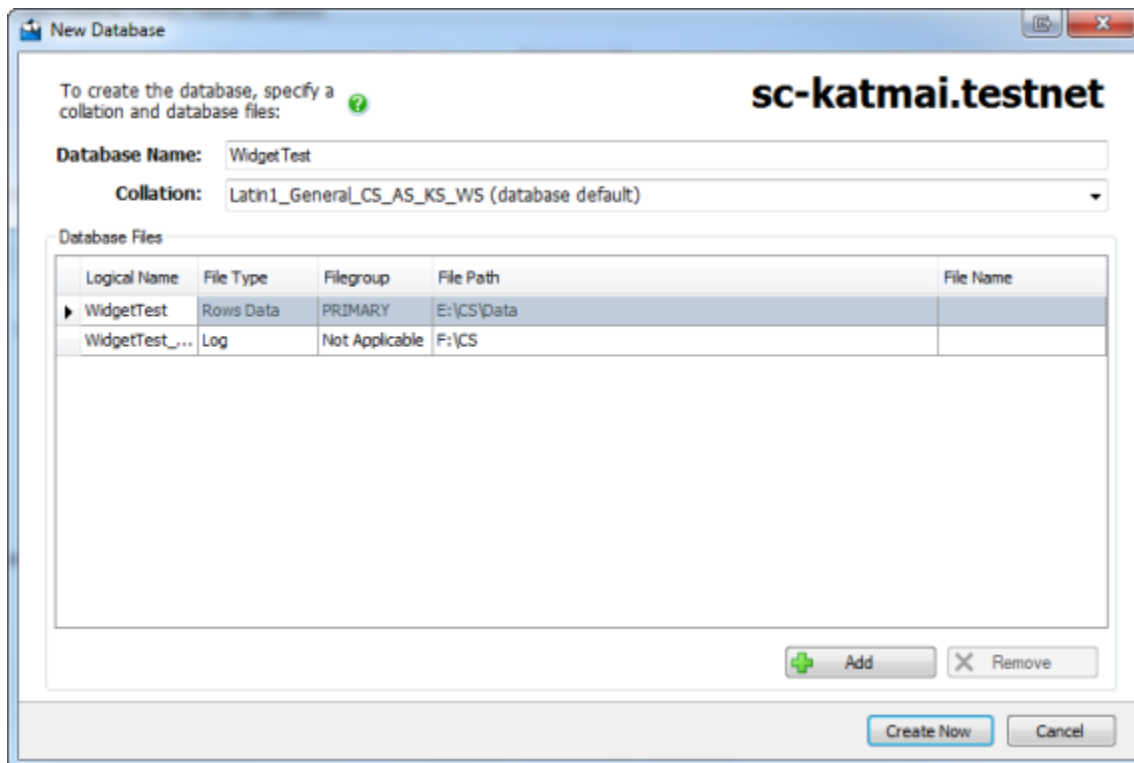
With SQL Compare 10, you can create a new database, which you can then select to compare. This can be useful if you want to create a copy of the source database.

To create a new database:

1. On the Project Configuration dialog box, select **Database** as a data source. Specify connection details for the server you want to create the database on. In the **Database** box, select **Create Database**:



The New Database dialog box is displayed:



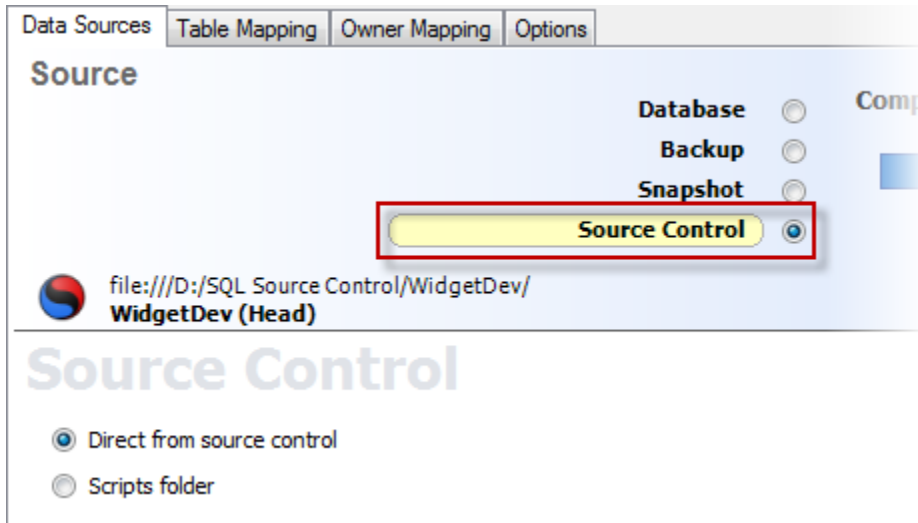
2. Specify a name, default collation, and database files for the new database. By default, SQL Compare adds the same database files from the source database to the new database.

- A primary data file and transaction log file are required when creating a database; you can't remove them
- If you're creating a database on SQL Azure, you can't add database files or change the collation

## Comparing versions from SQL Source Control

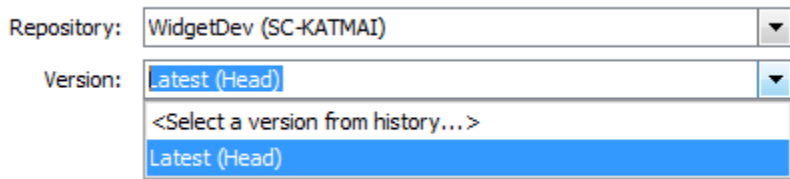
In SQL Compare 10, you can select specific versions of a source-controlled database to compare:

1. On the Project configuration dialog box, select **Source Control** as a data source, and then select **Direct from source control**:



2. In the **Repository** box, select a database linked to SQL Source Control, or click **<Browse source control...>** to specify a repository URL.

In the **Version** box, select a specific revision from the source control history, or select the latest revision:



## New command line switches

The following command line switches are new in SQL Compare 10:

### **/AbortOnWarnings**

Alias: */aow*

Specifies that SQL Compare won't perform a deployment if there are any serious deployment warnings.

If you use this switch and there are serious deployment warnings, exit code 61 is displayed.

### **/IgnoreSourceCaseSensitivity**

When you're creating a scripts folder using */makescripts*, SQL Compare automatically detects the case sensitivity of the data source.

Use */ignoreSourceCaseSensitivity* to disable automatic detection of case sensitivity.

### **/LogLevel:<level>**

Alias: */log*

Creates a log file with a specified minimum log level.

Log files collect information about the application while you're using it. These files are useful to us if you encounter a problem. For more information, see [Logging and log files](#).

## Arguments:

• <i>None</i>	Disables logging
• <i>Error</i>	Reports serious and fatal errors
• <i>Warning</i>	Reports warning and error messages
• <i>Verbose</i>	Reports all messages in the log file

The default is None.

For example:

```
sqlcompare /db1:WidgetStaging /MakeScripts: D:\Scripts Folder
          /LogLevel:Verbose
```

You must use */LogLevel* each time you want a log file to be created.

## **/ShowWarnings**

Alias: */warn*

Displays any warnings that apply to the deployment.

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction
          /ShowWarnings
```

## **/Version1:<version1>**

Alias: */v1*

Specifies the source control version of the first (source) database. To specify a version, the database must be linked to SQL Source Control.

To specify the latest version, type: *HEAD*

The following example compares version 3 of WidgetStaging with the latest version of WidgetProduction:

```
sqlcompare /db1:WidgetStaging /version1:3
          /db2:WidgetProduction /version2:HEAD
```

## **/Version2:<version2>**

Alias: */v2*

Specifies the source control version of the second (target) database. To specify a version, the database must be linked to SQL Source Control.

## **/VersionUserName1:<versionusername1>**

Alias: */vu1*

Specifies the username for the source control server linked to the first (source) database.



```
sqlcompare /db1:WidgetStaging /v1:3 /versionusername1:User1 /vp1:P@ssw0rd  
/db2:WidgetProduction /v2:HEAD /versionusername2:User2 /vp2:Pa$$w0rd
```

If you have a username saved in SQL Source Control, you don't need to specify it in the command line.

### **/VersionUserName2:<versionusername2>**

Alias: */vu2*

Specifies the username for the source control server linked to the second (target) database.

### **/VersionPassword1:<versionpassword1>**

Alias: */vp1*

Specifies the password for the source control server linked to the first (source) database.

```
sqlcompare /db1:WidgetStaging /v1:3 /vu1:User1 /versionpassword1:P@ssw0rd  
/db2:WidgetProduction /v2:HEAD /vu2:User2 /versionpassword2:Pa$$w0rd
```

If you have a password saved in SQL Source Control, you don't need to specify it in the command line.

### **/VersionPassword2:<versionpassword2>**

Alias: */vp2*

Specifies the password for the source control server linked to the second (target) database.

## **New project options**

The following project options are new in SQL Compare 10:

### **Add object existence checks**

When this option is selected, SQL Compare checks for the existence of objects affected by the deployment by adding IF EXISTS statements in the deployment script.

This option can be useful if you want to run the deployment script multiple times.

### **Use DROP and CREATE instead of ALTER**

When this option is selected, SQL Compare replaces ALTER statements in the deployment script with DROP and CREATE statements for the following objects:

- Views
- Stored Procedures
- Functions
- Extended Properties
- DDL Triggers
- DML Triggers

If you select this option, you must also select the **Add object existence checks option**, or the deployment script will fail.

## Ignore migration scripts for databases

When this option is selected, SQL Compare won't try to retrieve migration scripts when you compare a database.

(By default, when you compare a database that has an associated revision number, SQL Compare tries to connect to source control to retrieve any relevant migration scripts.)

This option can be useful if you've encountered an error connecting to source control when comparing a database.

## Connecting to SQL Server 2012

SQL Compare 10 can connect to SQL Server 2012 (Denali) servers.

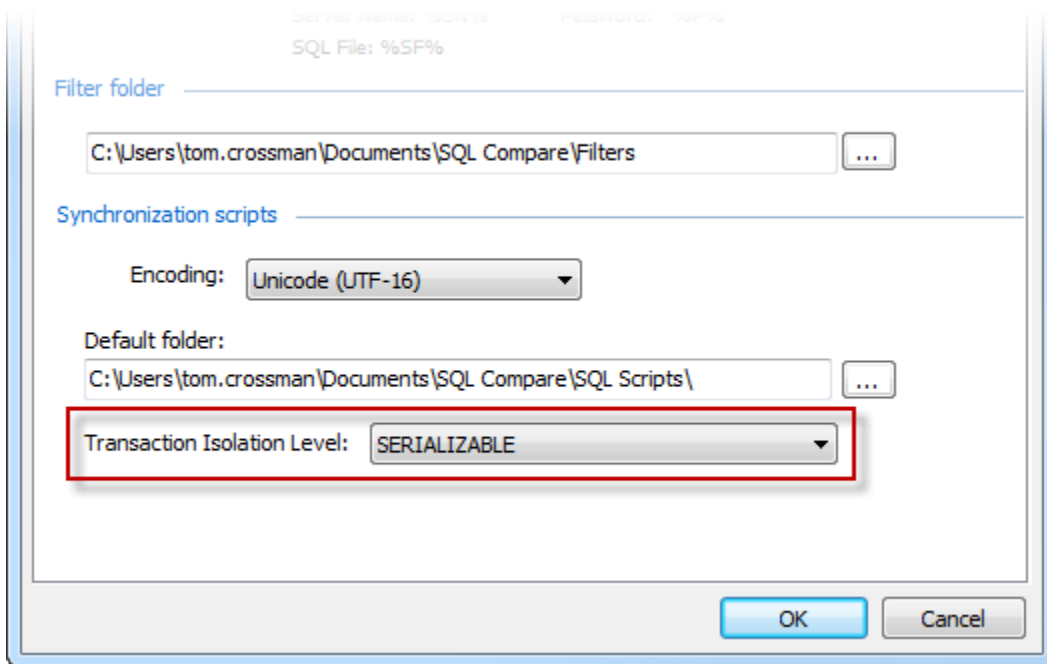
If you have any feedback on the new SQL Server 2012 features you want SQL Compare to support, [let us know](#).

## Transaction Isolation Level application option

SQL Compare 10 includes a new option you can use to set the transaction isolation level used in deployment scripts. This option is useful to prevent deployment errors when using linked servers.

To set the transaction isolation level:

1. In the SQL Compare **Tools** menu, click **Application Options**.
2. Under Deployment scripts, click the **Transaction Isolation Level** box, and then select the level you want to set:



# About SQL Compare

With SQL Compare, you can compare and deploy the structure of two Microsoft SQL Server databases.

This is useful, for example, in a development environment when changes made to a local database need to be transferred to a live database on a remote server. Traditionally, this meant spending hours scrutinizing the database structure and hand-generating deployment scripts. SQL Compare automates this process for you.

You can use SQL Compare to compare and deploy SQL Server 2012, SQL Server 2008, SQL Server 2005, and [SQL Azure](#) databases.

## Quick start guide

1. [Choose the data sources to compare](#)
2. [View the comparison results](#)
3. [Set up the deployment](#)
4. [Create the deployment script](#)

## Worked examples

Learn more about SQL Compare by following these detailed examples:

- [Comparing and deploying two databases](#)
- [Using SQL scripts as a data source](#) (requires SQL Compare Professional edition)
- [Deploying with migration scripts](#) (requires SQL Source Control)

## Third-party software used by SQL Compare

SQL Compare uses:

- [SQL LocalDB Wrapper](#) (distributed under the [Apache license](#))
- [Task Parallel Library](#)
- [SQLite](#)

# 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](mailto: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](https://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](#)

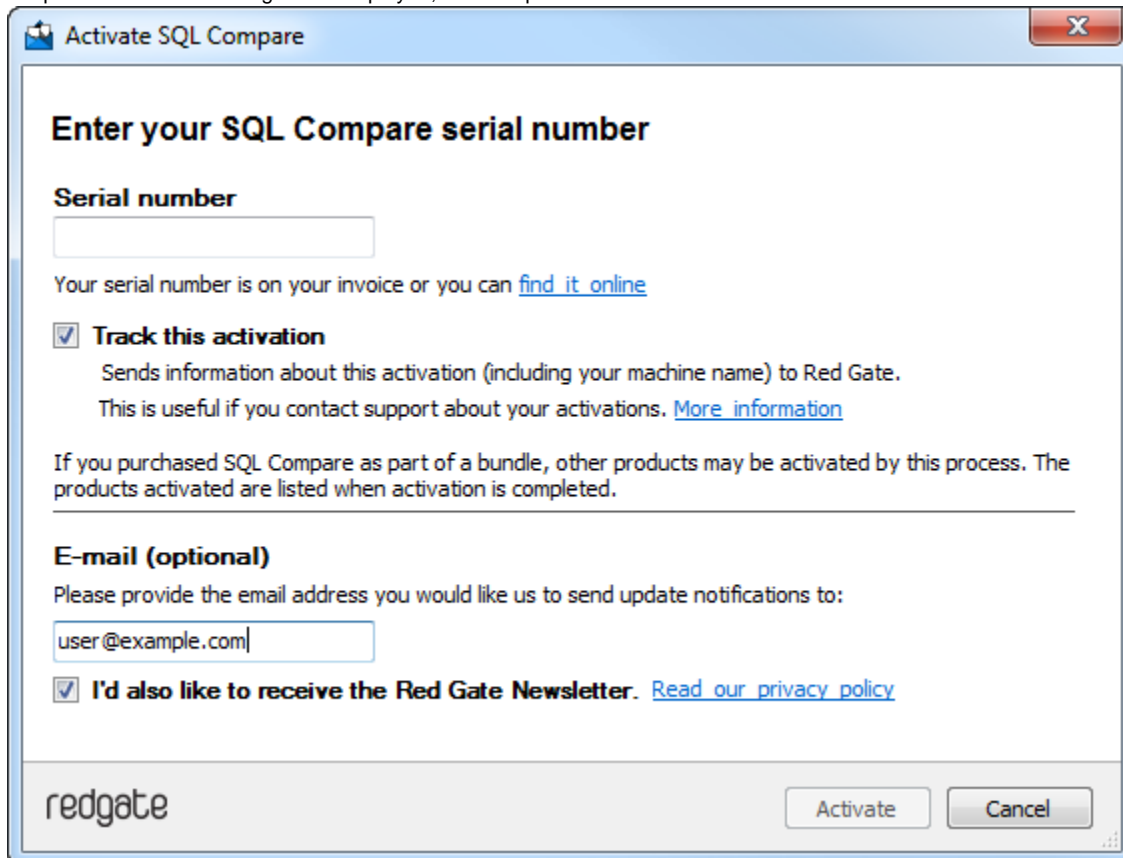
## Activating using the GUI

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:



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

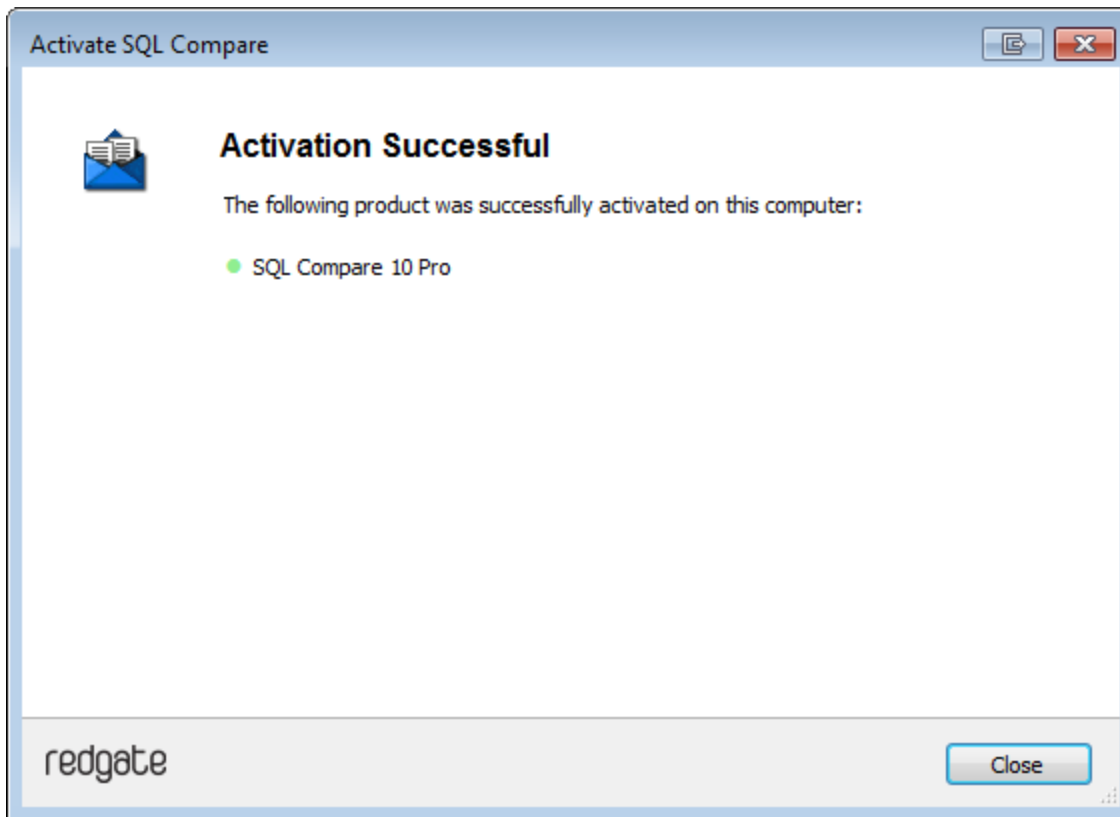
**I'd also like to receive the Red Gate Newsletter.** [Read our privacy policy](#)

redgate

Activate Cancel

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:





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:

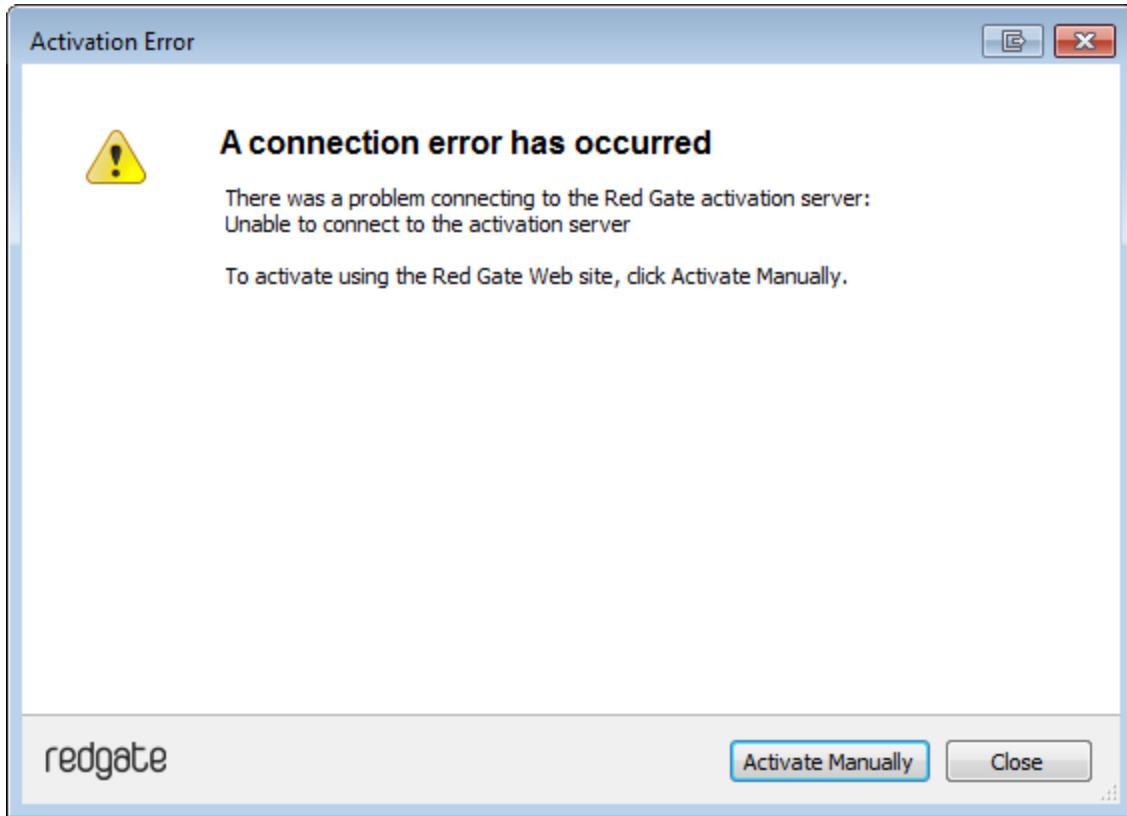
```
sqlcompare /activateSerial:123-456-789012-ABCD
```

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.

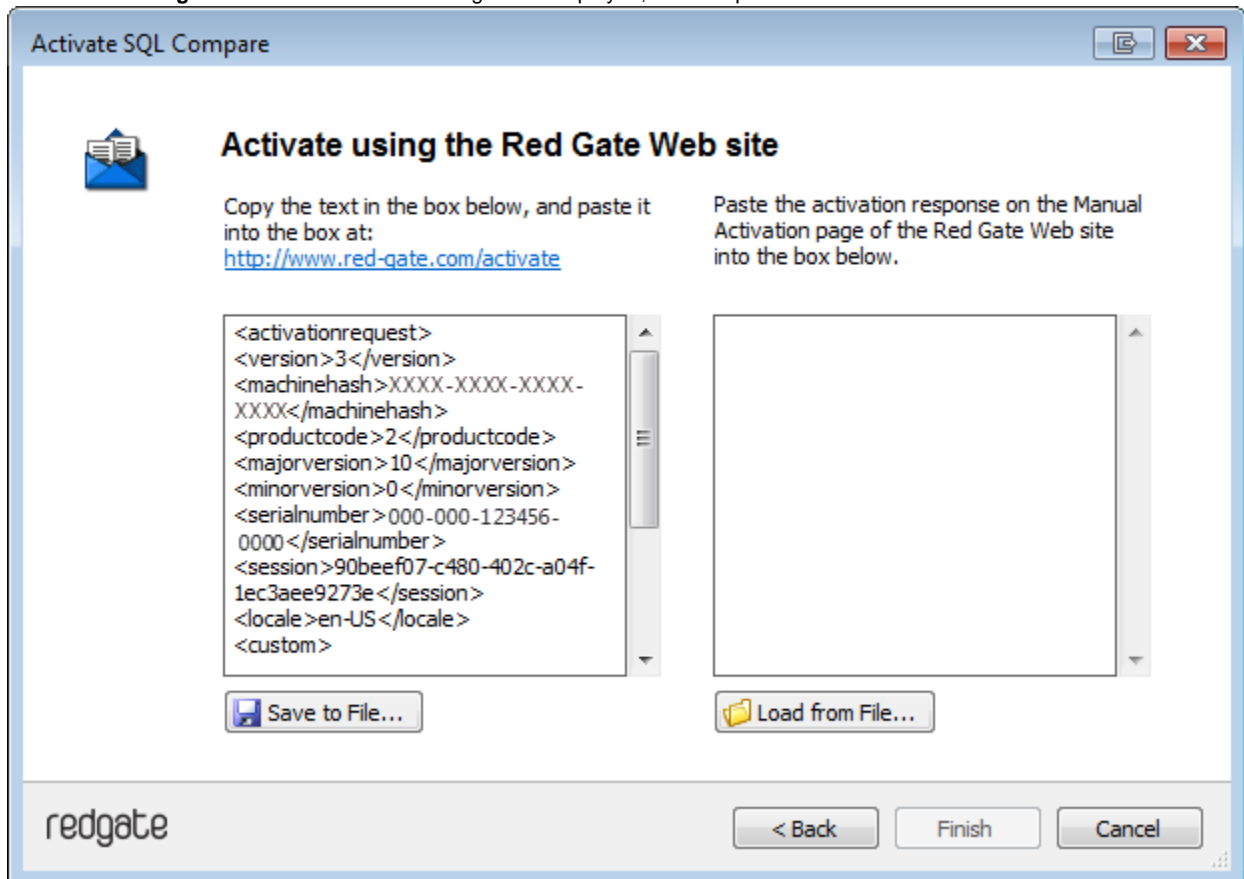
You can use manual activation whenever the **Activation Error** dialog box is displayed and the **Activate Manually** button is available, for example:



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:



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**.

Account ▾ Quotes Shopping Cart

redgate  
ingeniously simple tools

Home Products Store Community Support Our Company

I'm looking for... 🔍

## Manual Activation

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

### 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>90bef07-c480-402c-a04f-1ec3aee9273e</session>
<locale>en-US</locale>
<custom>
```

Get Activation Response

### Step 2

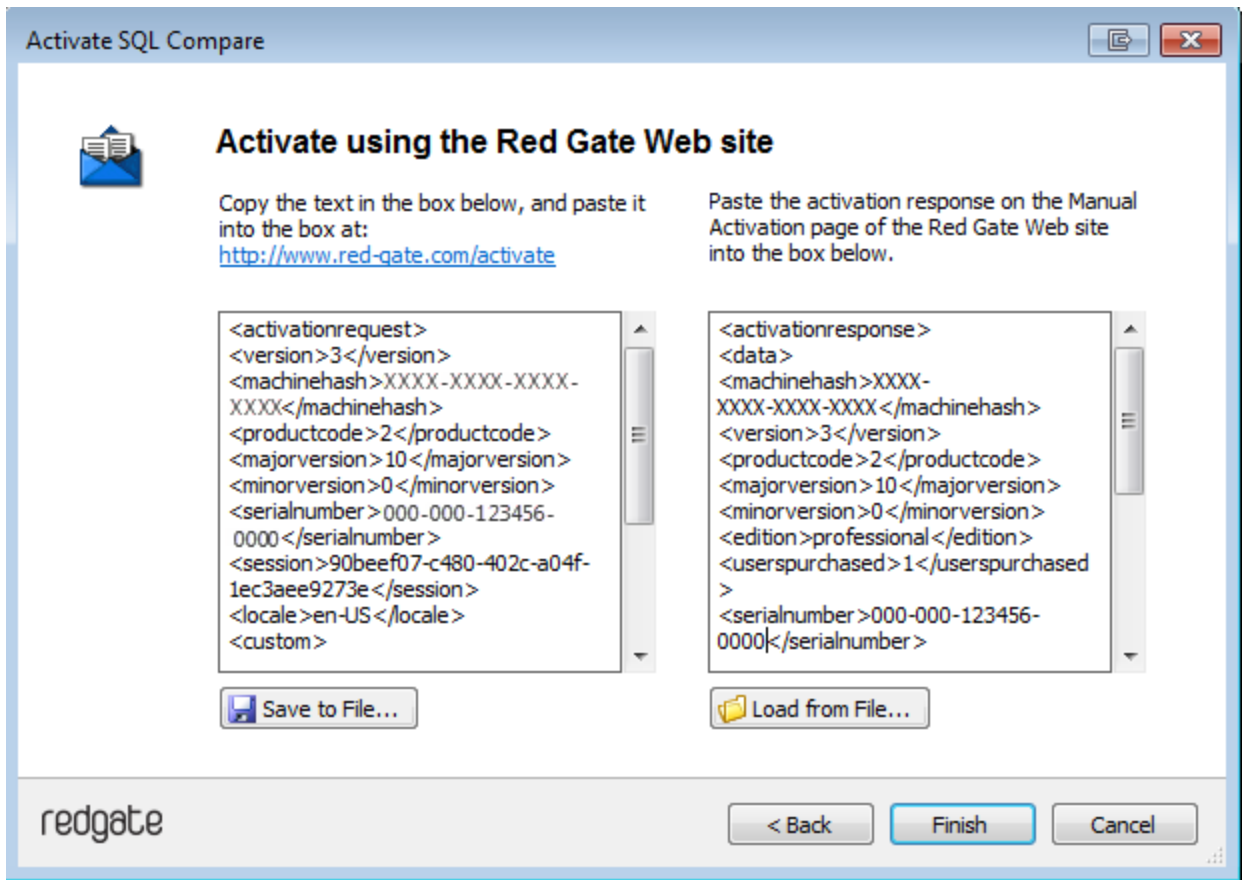
Copy the contents of this box into your product activation dialog box.

Save to File...

### Got a question?

0800 169 7433  
shop@red-gate.com


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.



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
```

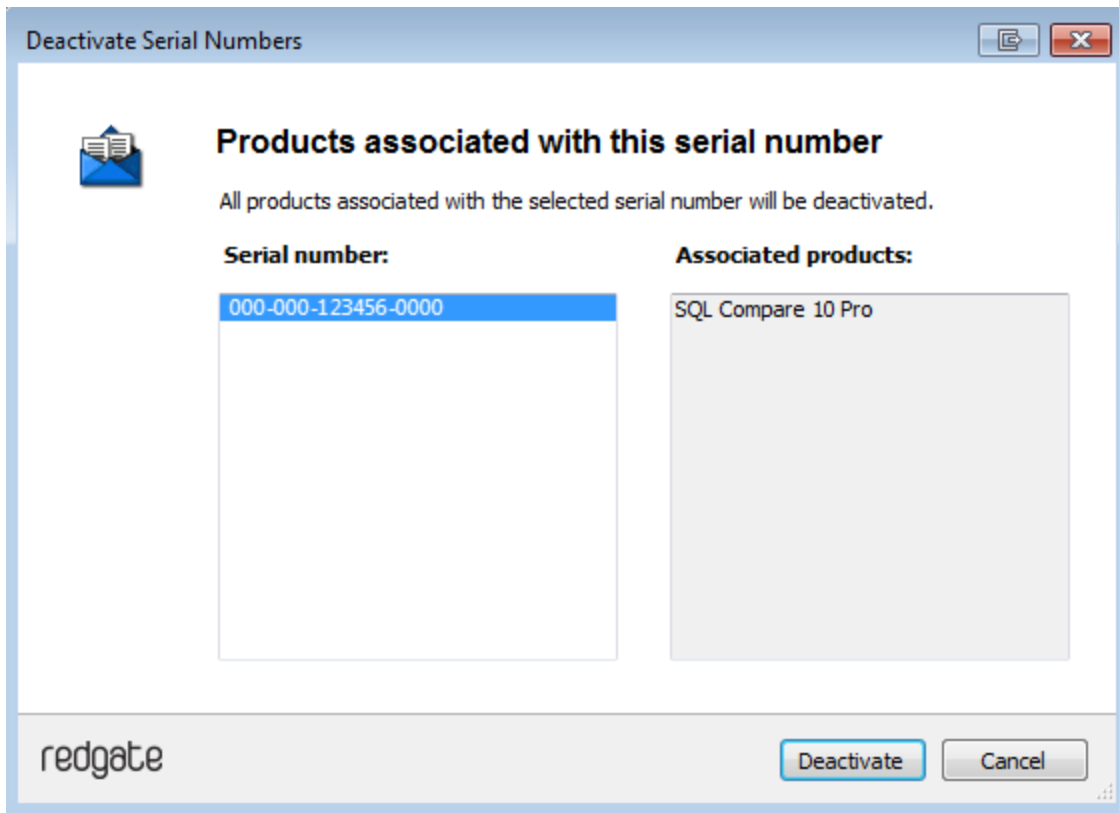
The **Deactivate Serial Numbers** dialog box is displayed. Follow the instructions below.

### 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:



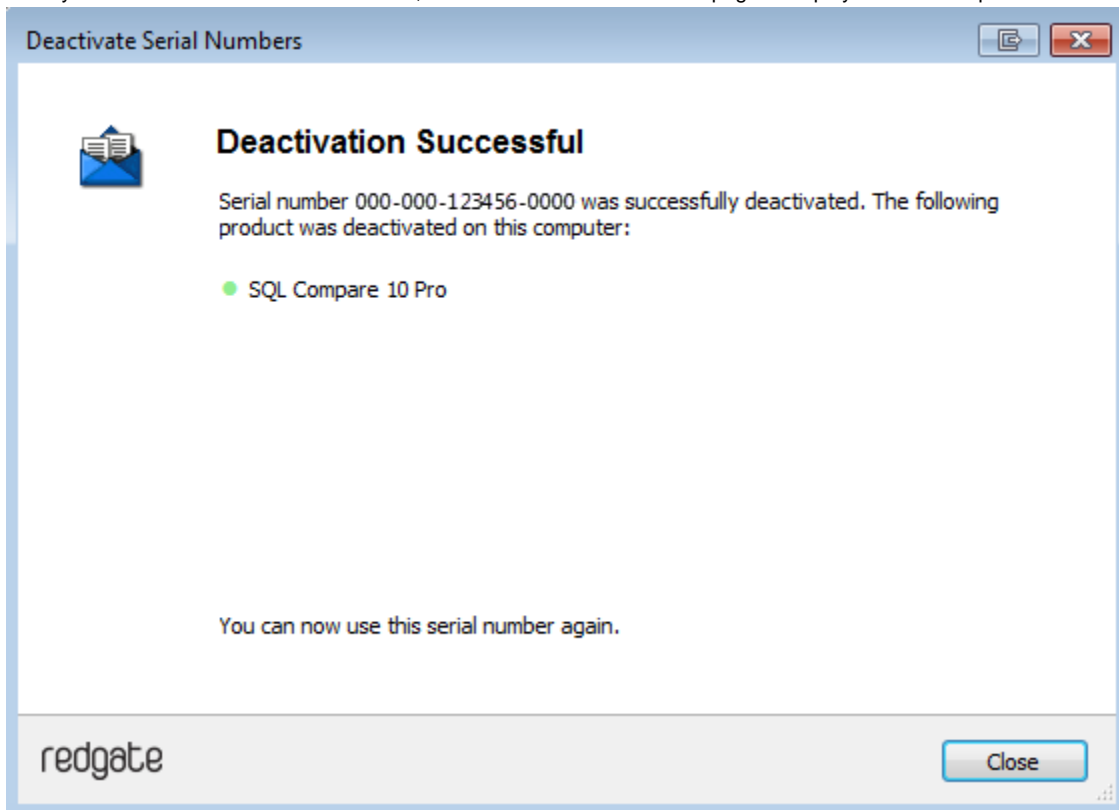
If you're running the executable file, the dialog box displays all the serial numbers for Red Gate products that have been activated on your computer.

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:



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](mailto: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](mailto: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.

## Changes to distribution of command line

From SQL Compare 10.2.0.1337, the licensing model for the SQL Compare command line has changed. You can now buy a DLM Automation Suite license which lets you use Redgate's SQL Server command line tools as part of an automated process. For more information, see [DLM Automation Suite](#).

The previous method of distributing the command line version of SQL Compare can no longer be used.

### How does this affect my licensing?

Under the [Comparison SDK license](#), you are allowed to distribute up to 10 copies of the command line version of SQL Compare with your own custom application.

If you are distributing the command line version to use with your own custom application (or you were previously distributing the command line), **please contact our Sales team for assistance**. You can then obtain the new DLM Automation Suite that is required to run the distributed command line version of SQL Compare.

### Does that mean my Professional version of SQL Compare will no longer allow me to use the command line?

No. The SQL Compare Professional license still allows you to run the command line on your own machine. However, a DLM Automation Suite license is needed for deploying the command line to machines which are not your own, such as build servers.

### When running the command line version of SQL Compare, I see a message that the Automation license is a trial version. What does this mean?

This means that you are not licensed for Automation, which will prevent you from running the command line version of SQL Compare after the trial date ends, unless you have activated a Professional license for SQL Compare or a DLM Automation Suite license.

### How do I activate the new DLM Automation Suite license?

Once you have received the correct Automation license required, you can activate the license on the machine that will be running the command line version of SQL Compare. For more information, see [Activating](#).

This will generate the appropriate *.lic* file on the machine that will be running SQL Compare at command line.

### Which files are needed to distribute the command line version of SQL Compare?

From SQL Compare 10.2.0.1337, the following files are required to distribute the command line:

- *SQLCompare.exe*
- *SQLCompare.exe.config*
- *RedGate.SOCCompareInterface.dll*
- *RedGate.BackupReader.CryptoHelper.dll*
- *LinqBridge.dll*
- *LinqBridge-License.txt*
- *RedGate.SOCCompareInterface.dll*
- *System.Data.SQLite.dll* (which should be placed in the SQLite folder)
- *Zlib1.dll*

For information about earlier builds, see [Integrating the command line with applications](#).

# 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. When 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](mailto: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.

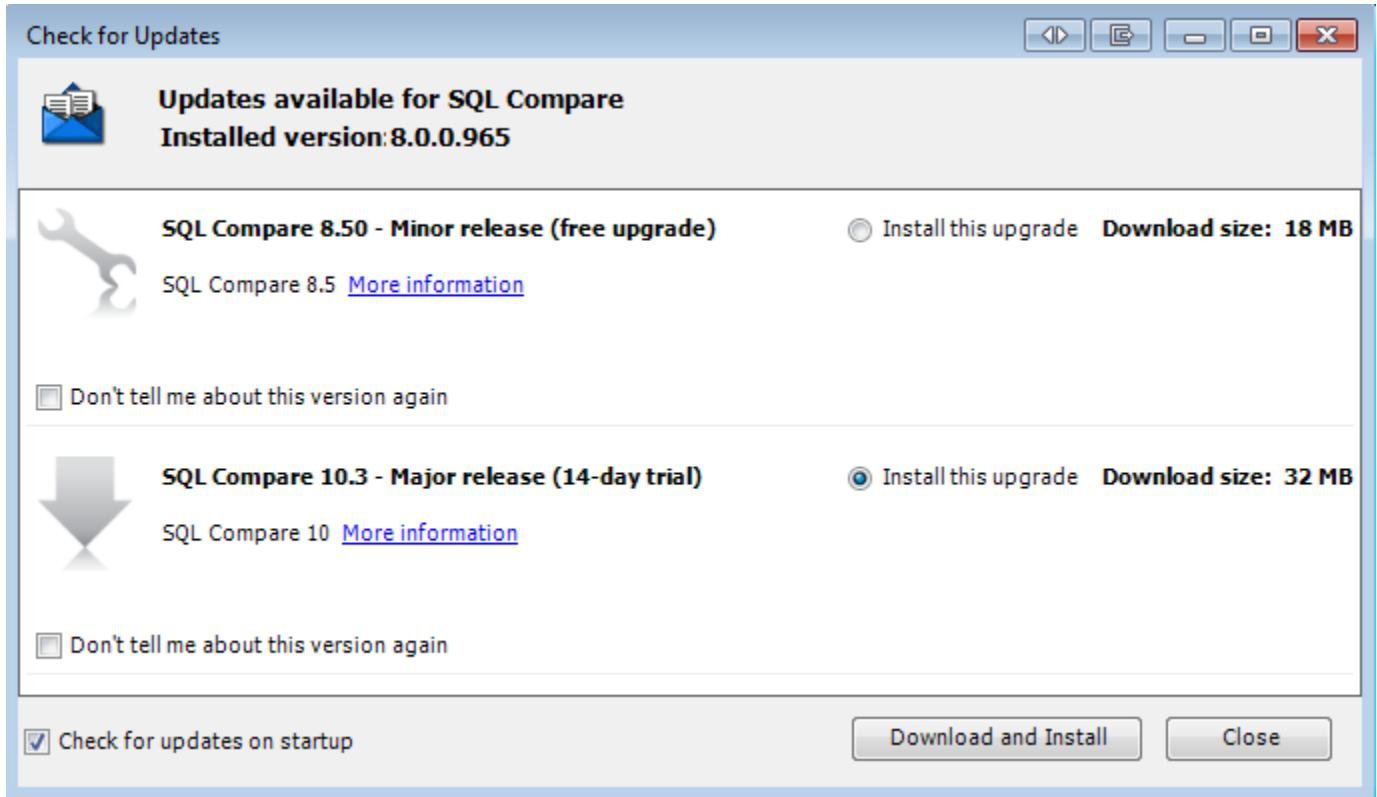
## Using Check for Updates

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:



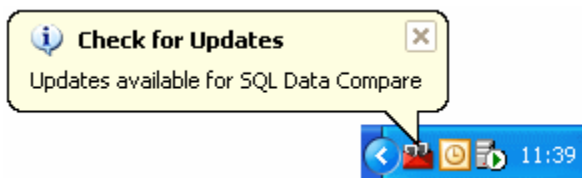
To view the full release details in your default web browser, click **More information**.

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:



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](#).

# Setting up the comparison

These pages explain how to set up a comparison in SQL Compare.

- [Working with projects](#)
- [Setting data sources](#)
- [Creating a new database](#)
- [Mapping tables and columns](#)
- [Mapping owners](#)
- [Setting project options](#)
- [Comparing databases on unconnected SQL Servers](#)
- [Permissions required to use SQL Compare](#)



## Working with projects

Whenever you compare databases, you set up a project. If you have any existing projects, the Project Configuration dialog box for your most recently used project is displayed when you start SQL Compare.

A project contains details of:

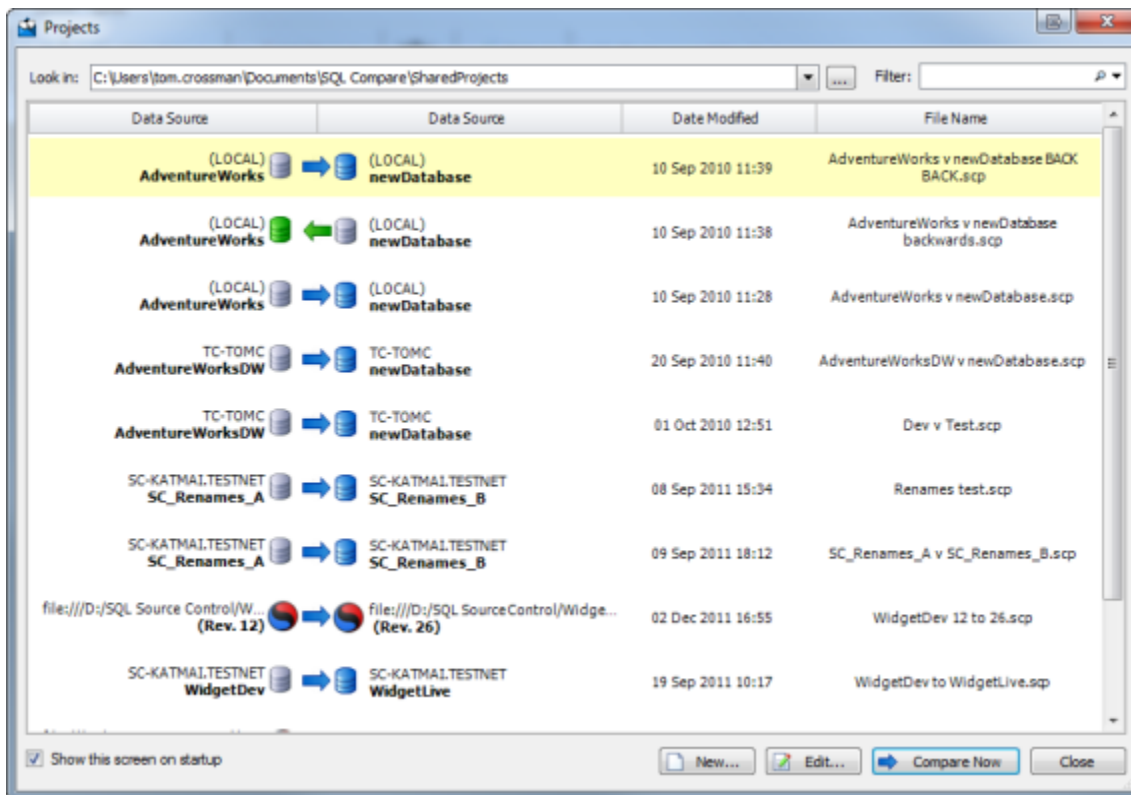
- which **data sources** you selected.  
If you selected a backup as a data source, the project saves details of the backup set you selected.
- the connection details for your data sources.
- which **project configuration options** you selected.
- which objects you have selected for deployment.
- your **owner mappings**.
- your **table mappings**.
- your most recently used **filter**.

## Finding a project

On the toolbar, click



(Open Project) to display the **Projects** dialog box:



The **Projects** dialog box shows details of your projects.

You can edit or compare these projects, or create a new project.

## Creating and editing a project

To create a new project, click



**New.**

To edit the current project, click



## **Edit.**

To open and edit an existing project, double click the project on the **Projects** dialog box.

## **Saving a project**

SQL Compare does not automatically save projects.

To save a project, on the Project Configuration dialog box click **Save**. Alternatively, you can save a project when you are reviewing its comparison results. To do this, on the **File** menu click **Save Project**.

If there are unsaved changes in the current project, you will be prompted to save when you create a new project, open another project, or when you close the application.

## **Copying a project**

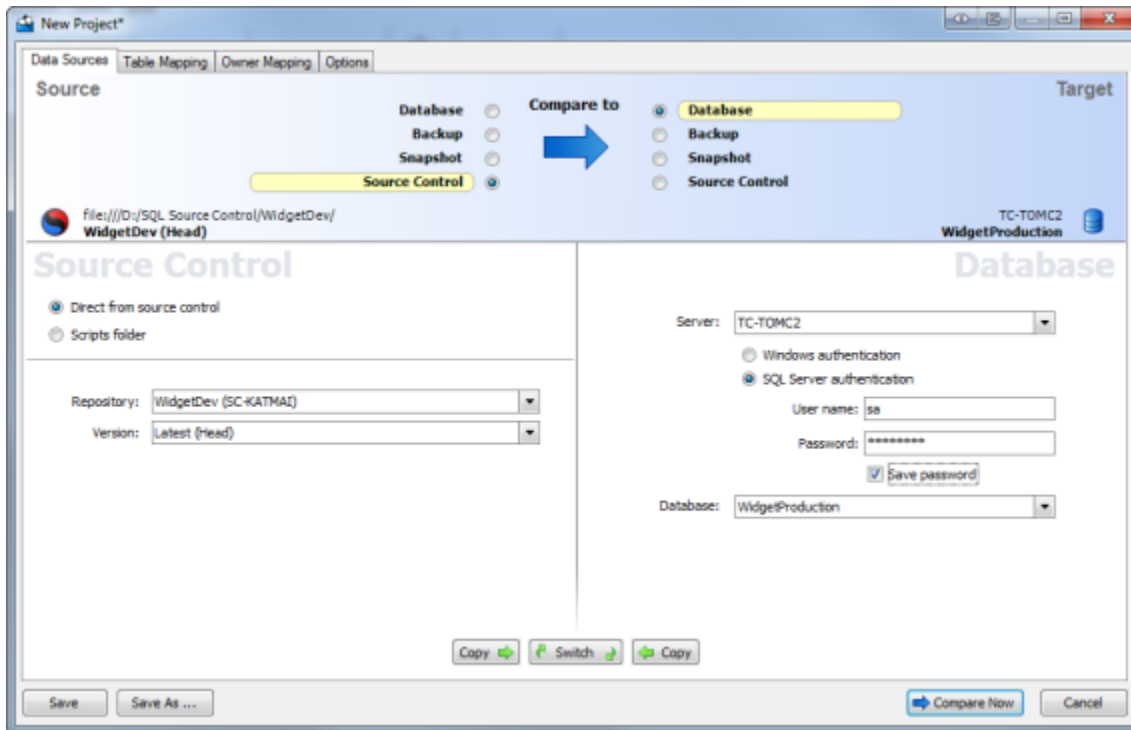
To make a copy of a project, on the **Projects** dialog box, select the project, right click, and select **Create Clone**.

Alternatively, open the project that you want to copy, and on the Project Configuration dialog box, click **Save As**.

To copy a project when you are reviewing the comparison results, on the **File** menu, click **Save Project As**.

## Setting data sources

When you create a new comparison project, SQL Compare requires information about which two data sources you want to compare, and how to connect to them. You specify this information on the Project Configuration dialog box:

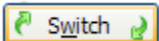


## Selecting data sources

Specify the two data sources you want to compare in the **Data Sources** tab. You specify a *source* and a *target*.

The *source* is the data source that will not change. The *target* is the data source that will change.

To switch the source and target, click



You can compare the following types of data source:

- Database  
A database you can connect to on a SQL Server. You will need to enter connection details.  
If you want to compare Oracle or MySQL databases, see: [Schema Compare for Oracle / MySQL Compare](#).
- Backup  
Native SQL Server backups or Red Gate SQL Backup backups. You can specify a backup as a data source only if you are using SQL Compare Professional edition.  
For more information, see: [Working with backups](#).
- Snapshot  
A file produced by SQL Compare that contains information about the structure of a database.  
For more information, see: [Working with snapshots](#).
- Source Control  
There are two options for the Source Control data source:
  - Direct from source control  
A revision of a database from a source control repository.  
For more information on getting your database into source control, see: [SQL Source Control documentation](#).
  - Scripts folder  
A folder containing the SQL creation script files that collectively define the database structure. This is usually organized into a number of subfolders.  
For more information, see: [Working with scripts folders](#).

You can specify Source Control as a data source only if you are using SQL Compare Professional edition.

You can compare any combination of data source types in your project.

If you select a snapshot or a backup as the target, the snapshot file or backup will not be modified. Instead, a deployment script is generated to

modify the database from which the snapshot or backup was made.

If you select a scripts folder as the target, you can choose to modify those script files directly during deployment, or to create a deployment script to modify the database from which the scripts folder was created.

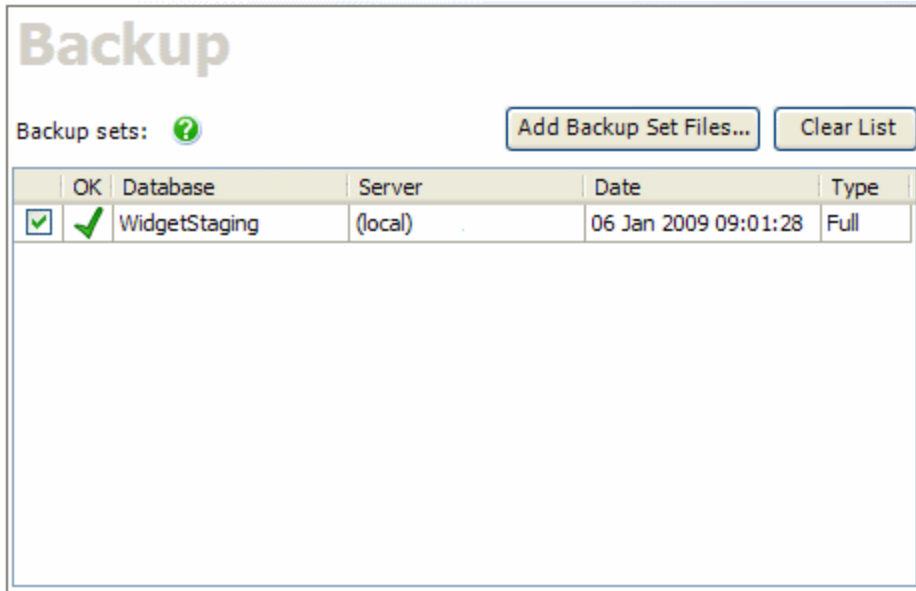
## Selecting a database

1. Under Source or Target, select **Database**.
2. Type or select the name of the SQL Server in the **Server** box.  
If you have 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**.  
If you want SQL Compare to remember your password, select the **Save password** check box.
4. In the **Database** box, type or select the name of the database.  
If you want to create a new database, at the top of the database list, click **Create Database**. For full details, see: [Creating a new database](#).

To refresh the **Database** list, right-click the box and click **Refresh**, or scroll to the top of the list and click **Refresh**.

## Selecting a backup

1. Under Source or Target, select **Backup**.



2. Click **Add Backup Set Files** to select all the files making up the backup set you want to compare.

To specify a network path, type the full path, including the server name, for example:

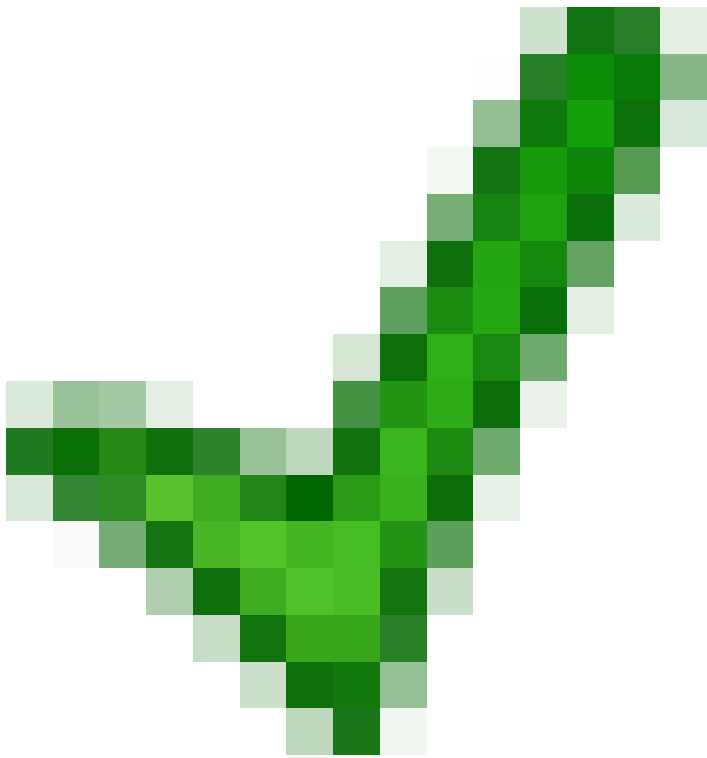
```
\\ServerName\MyFolder\MyFile
```

If any of the files you add are encrypted, the **Decrypt Backup Files** dialog box is displayed. Enter the password to decrypt these files.

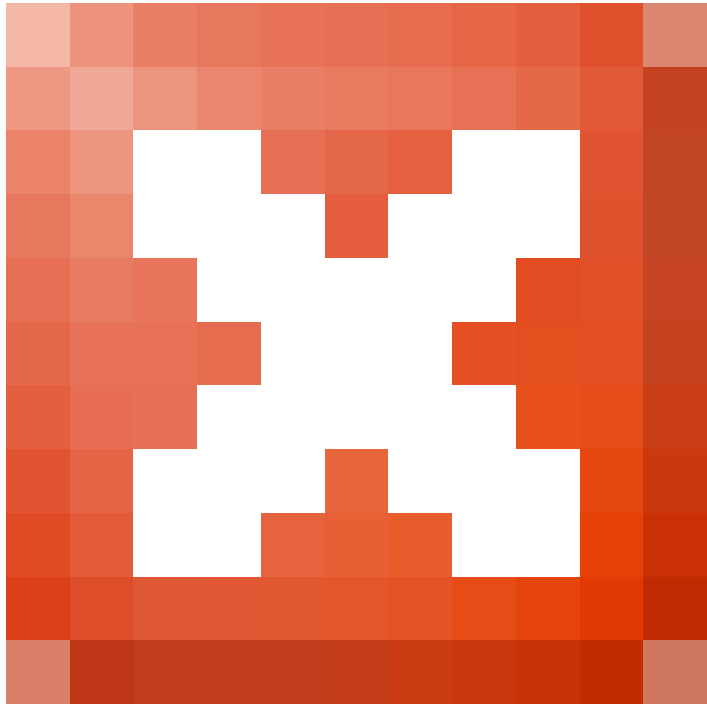
To use a differential backup as a data source, you must also add the associated full backup.

SQL Compare does not support using partial, filegroup, or transaction log backups as a data source.

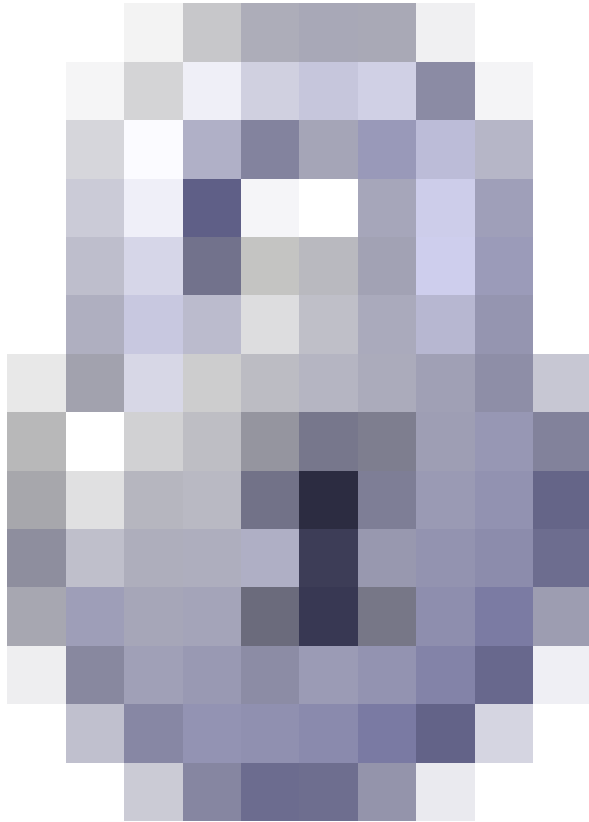
When you have added a backup set, one of the following icons is displayed:



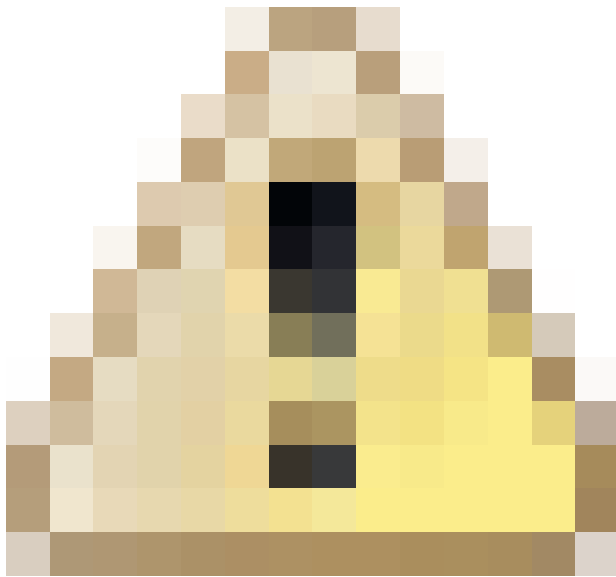
The backup set is valid and complete.  
Select the check box to use this backup as a data source.



The backup set you have selected cannot be used as a data source.  
This error is shown if the backup is corrupted, or if you have selected a partial, filegroup, or transaction log backup.



One or more files in the backup set is encrypted.  
Click the padlock icon to display the **Decrypt Backup Files** dialog box.



Either the backup set is incomplete, or a differential backup has been added without the corresponding full backup.

## Selecting a SQL Compare snapshot

1. Under Source or Target, select **Snapshot**.
2. In the **Snapshot** box, select the name of the snapshot, or click

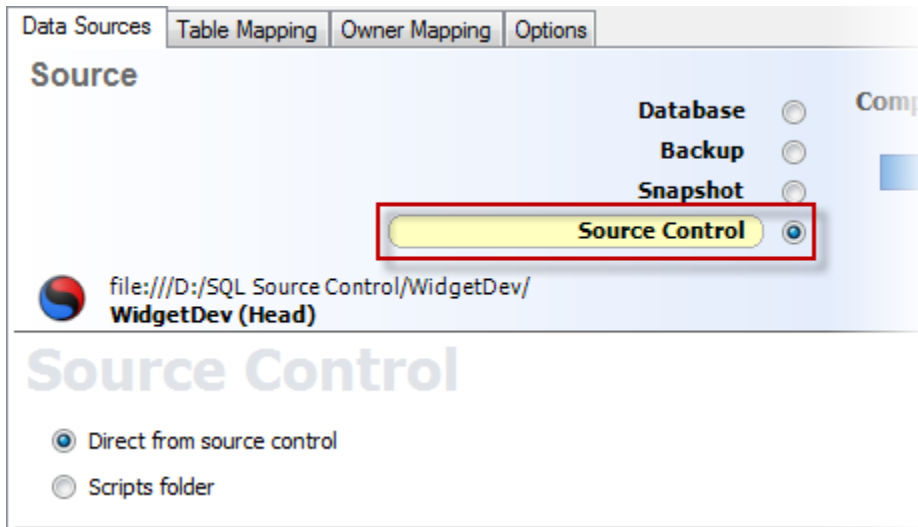


to browse to the snapshot file.

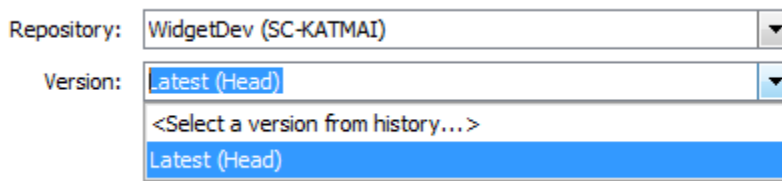
SQL Compare displays information about the database from which the snapshot was created.

## Selecting a version from source control

1. Under Source or Target, select **Source Control**, and then select **Direct from source control**:



2. In the **Repository** box, select a database linked to SQL Source Control, or click **<Browse source control...>** to specify a repository URL.
3. In the **Version** box, select a specific version from the source control history, or select the latest version:



## Selecting a scripts folder

1. Under Source or Target, select **Source Control**, and then select **Scripts folder**.
2. In the **Scripts Folder** box, select the folder, or click



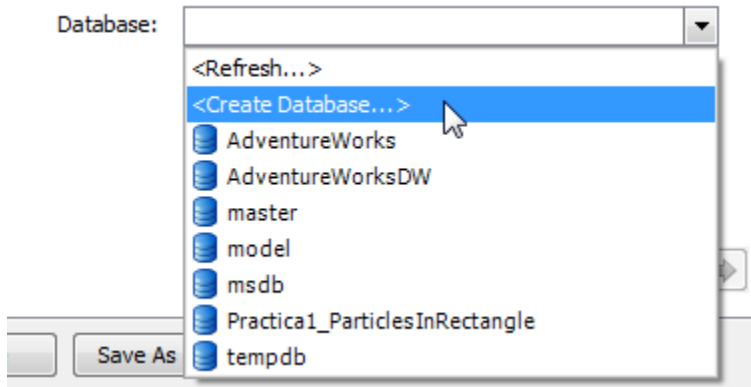
to browse to the folder.

## Creating a new database

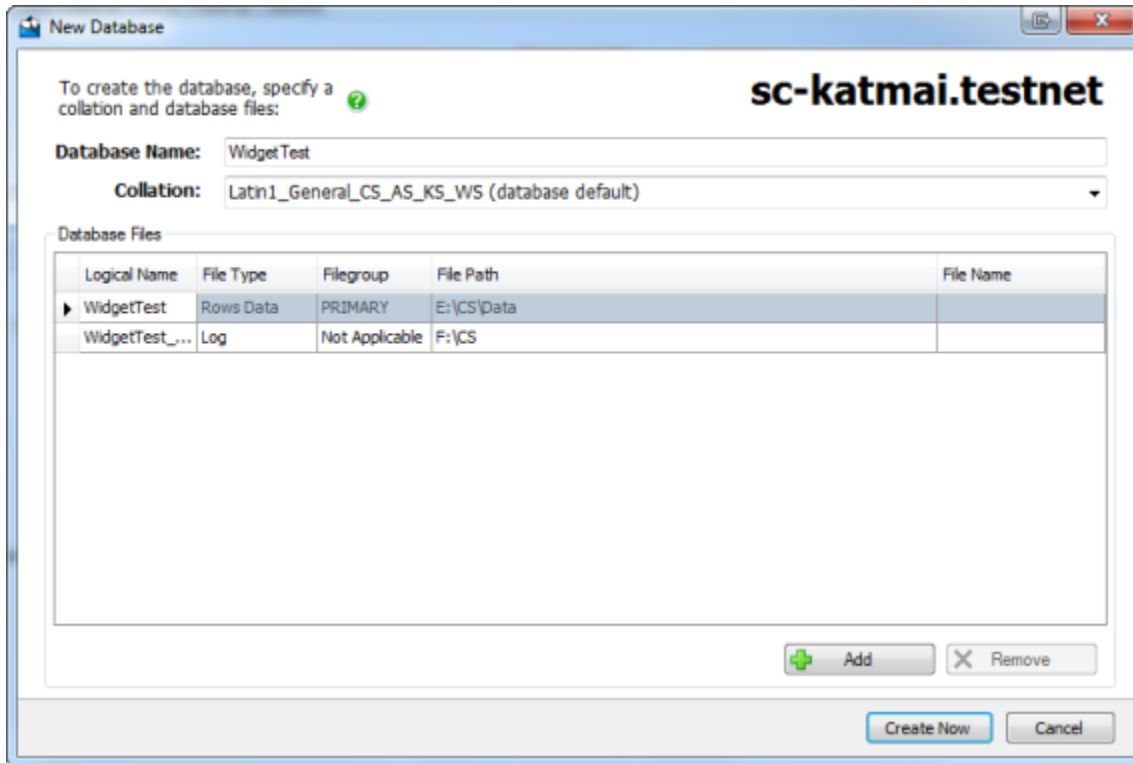
When you're setting data sources, SQL Compare can create a new database which you can then select to compare. This can be useful if you want to create a copy of the source database.

To create a new database in SQL Compare:

1. On the Project Configuration dialog box, select **Database** as a data source.
2. Specify connection details for the server you want to create the database on.
3. In the **Database** box, select **Create Database**:



The **New Database** dialog box is displayed:



4. Specify a name, default collation, and database files for the new database.  
By default, SQL Compare adds the same database files from the source database to the new database.
5. Click **Create Now**.

- A primary data file and transaction log file are required when creating a database; you can't remove them.
- If you are creating a database on SQL Azure, you can't add database files or change the collation.



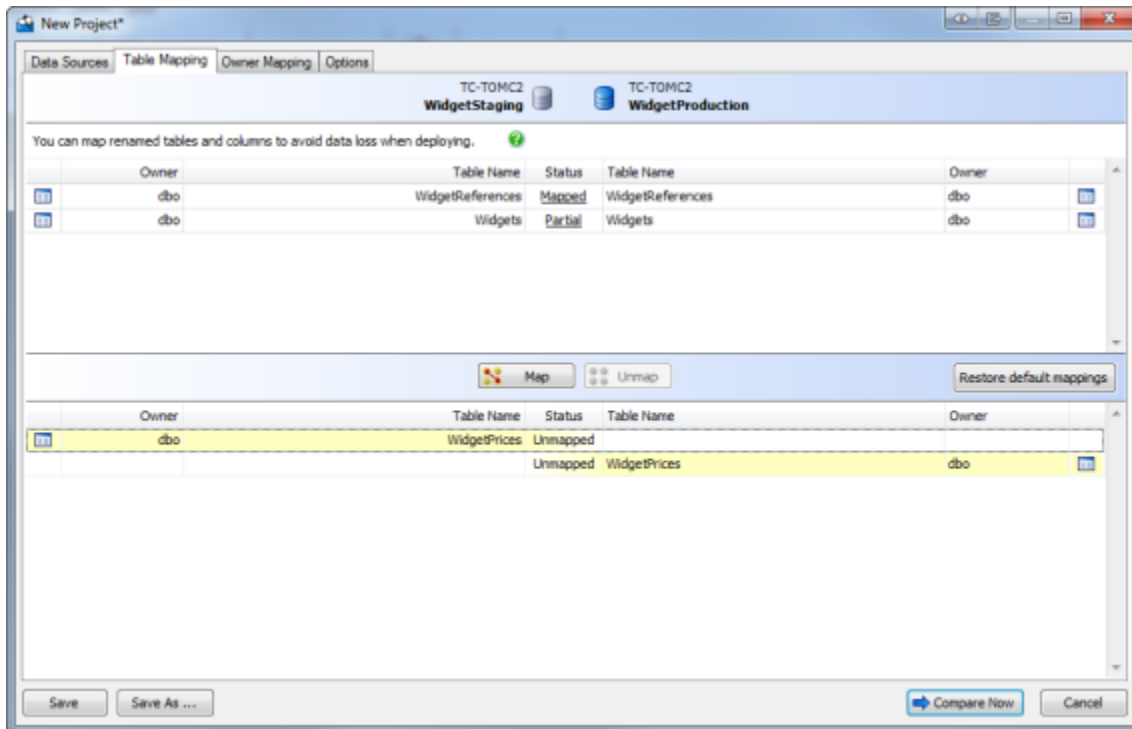
## Mapping tables and columns

When you have selected your data sources, SQL Compare automatically maps tables and columns with the same name in the source and target data sources.

However, SQL Compare also allows you to map together tables and columns with different names. This can be useful to prevent data loss when deploying tables or columns that have been renamed.

For example, if you map the table *TableA* in the source to the same table that has been renamed as *TableB* in the target, SQL Compare will compare the table as an object that exists in both data sources. When you deploy the table, the name change will be scripted using the *sp\_renam*e system stored procedure; the table is not dropped and re-created.

To compare tables and columns that are not automatically mapped, click the **Table Mapping** tab of the Project Configuration dialog box:



The upper pane displays tables that are fully **Mapped** or have **Partial** mapping. The lower pane displays Unmapped tables.

To set the column mappings for a table, click the **Status** link for the object you want to re-map.

If a table has a **Partial** mapping, some of its columns are not mapped, and cannot be compared.

## Mapping tables

To map tables:

1. On the **Table Mapping** tab, select an *Unmapped* table that you want to map from the *source* database.
2. Select the *Unmapped* table that you want to map from the *target* database.
3. Click



**Map.**

SQL Compare moves the tables to the upper pane.

To unmap tables and views:

1. On the **Table mapping** tab, select a *Mapped* table, or a *Partial* mapping.
2. Click



**Unmap.**

SQL Compare moves the tables or views to the lower pane.

## Mapping columns

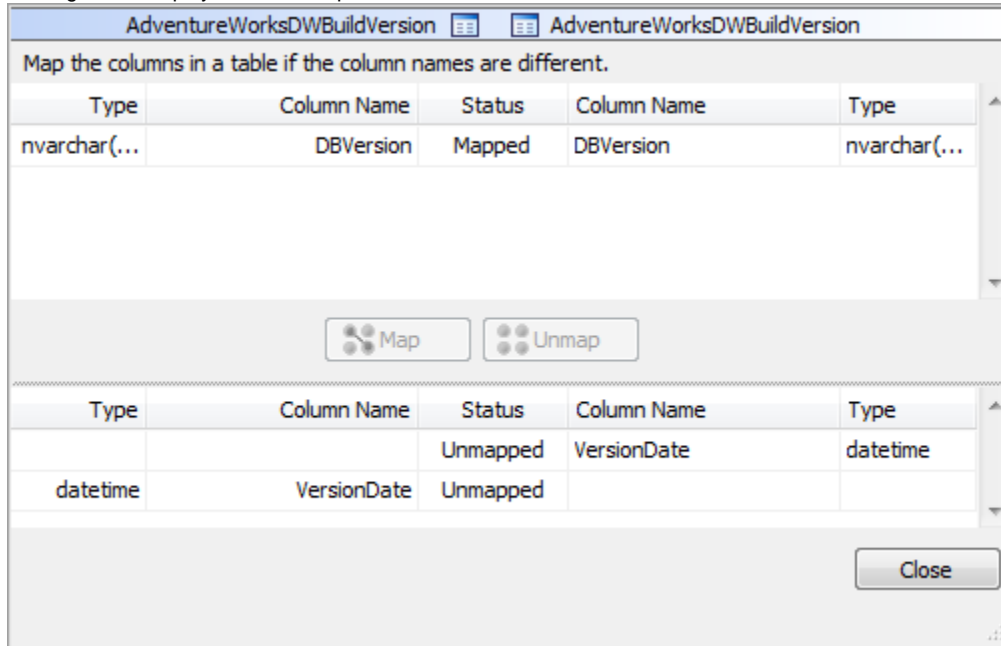
If you want to compare columns in a table and the column names are different, you can map the columns as required.


SQL Compare automatically maps columns with different names that are:

- the same data type
- the same position in the column order


To map columns:

1. On the **Table mapping** tab, click the **Status** box of a *Partial* mapping.  
A dialog box is displayed. For example:



2. Select an *Unmapped* column that you want to map from the *source* database.
3. Select an *Unmapped* column that you want to map from the *target* database.
4. Click  **Map**.  
SQL Compare moves the columns to the upper pane of the dialog box.
5. Click **Close**.

To unmap columns:

1. On the **Table mapping** tab, click the **Status** box of a *Mapped* table or view, or a *Partial* mapping.  
A dialog box is displayed for you to specify the column mappings, as shown above.
2. Select a *Mapped* column.
3. Click  **Unmap**.  
SQL Compare moves the columns to the lower pane of the dialog box.
4. Click **Close**.

## Mapping owners

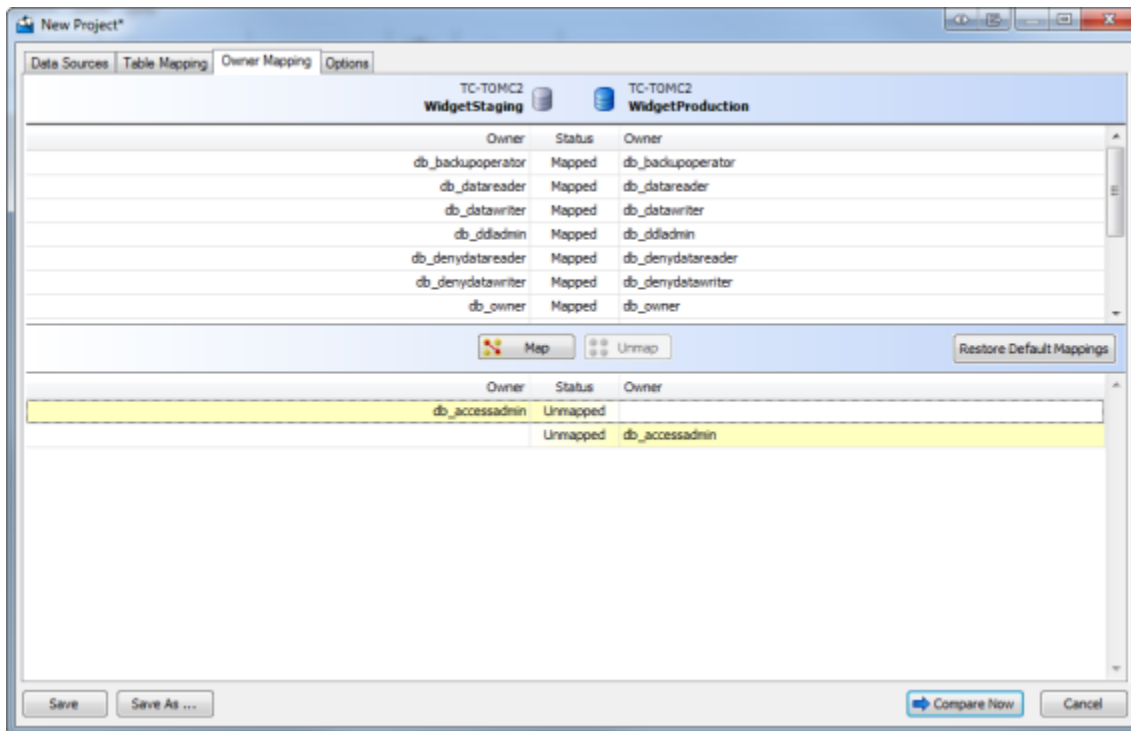
When you create a comparison [project](#) and you have selected your [data sources](#), SQL Compare automatically maps objects with the same name and the same owner (SQL Server 2000) or schema (SQL Server 2008 and SQL Server 2005) for you.

If you want to compare objects with the same name, but those objects have different owners or belong to different schemas, you can map the owners or schemas as required. For example, if you want to compare objects in a test database that are owned by *sales* with objects in a production database that are owned by *support*, you can map *sales* to *support*.

- If you map the owners (or schemas) *sales* and *support*, this enables SQL Compare to compare the stored procedures [sales].[editCustomerDetails] and [support].[editCustomerDetails]. SQL Compare can then rename other objects that are referenced by the stored procedure. For example, if the [support].[editCustomerDetails] stored procedure references the table [support].[customerDetails], the stored procedure is updated to reference the table [sales].[customerDetails]. The same applies to functions, DML triggers, DDL triggers, views, defaults, and rules.
- If you are comparing stored procedures, functions, DML triggers, views, defaults, or rules that contain WITH elements, they are flagged as different unless the **Ignore WITH element order** option is selected in your comparison project.
- You cannot perform the comparison if an existing schema or owner is not mapped. For example, if you unmap the schemas *Sales* and *Support*, and then map *Support* to *Sales*, the comparison will fail if you do not also map *Sales* to another schema.

## Editing the owner mappings

The **Owner Mapping** tab of the Project Configuration dialog box enables you to edit mappings:



The upper pane displays a list of schemas, database roles, and database users that SQL Compare has automatically mapped for you; the lower pane displays a list of schemas, database roles, and database users that are not mapped.

- To undo a mapping, in the upper pane, select the mapping, and click



**Unmap.**

- To create a mapping, in the lower pane, select item A from the *source*, then select item B from the *target*, and click



**Map.**

Next, select item B from the *source*, then select item A from the *target*, and click



**Map.**

For example, to map *sales* to *support*, you create a mapping for *sales* to *support* and then you create a mapping for *support* to *sales*.

To reset the mappings to the defaults, click **Restore Default Mappings**.

## Setting project options

Project options let you modify the behavior of SQL Compare. For example, you can set SQL Compare to ignore certain objects even if they're different, or so it doesn't script certain properties for deployment (such as the collation order on columns).

When you create a new project, you should run the comparison with the default options, then review your comparison results. However, if your database is on a SQL Server with case-sensitive sort order, you must select the *Treat items as case sensitive* option. When you have reviewed your comparison results, you may want to consider changing some of the options.

The options you set are saved for each project, and are modified on the Project Configuration dialog box.

To display the Project Configuration dialog box, click



(Edit Project), or select **Project Options** from the Tools menu.

Some of the options apply only to the comparison, and don't affect the deployment. Similarly, some options apply only to the deployment.

## Default options

To make the current selection of options your defaults, click **Save As My Defaults**. These saved defaults will be used for all new projects. To restore your defaults after making changes, click **My Defaults**.

To reset all the options to their original settings, click **Red Gate Defaults**. The default options for a project are as follows:

- Decrypt encrypted objects on 2005 and 2008 databases
- Ignore white space
- Ignore fill factor and index padding
- Ignore filegroups, partition schemes, and partition functions
- Ignore user properties
- Ignore WITH element order
- Ignore database and server names in synonyms

To search for an option, type search text in the **Find** box. The list is filtered to display only those options that contain the search text.

## Add object existence checks

When this option is selected, SQL Compare checks for the existence of objects affected by the deployment by adding IF EXISTS statements in the deployment script.

This option can be useful, for example, if you want to run the deployment script multiple times.

## Use DROP and CREATE instead of ALTER

When this option is selected, SQL Compare replaces ALTER statements in the deployment script with DROP and CREATE statements for the following objects:

- Views
- Stored Procedures
- Functions
- Extended Properties
- DDL Triggers
- DML Triggers

If you select this option, you must also select the **Add object existence checks option**, or the deployment script will fail.

## Enable fast deployment

This option allows you to deploy a database from the Project Configuration dialog box, bypassing the comparison results and deployment wizard.

When this option is selected, click the Deploy Now button to deploy a database.

By default, all objects are deployed. If you want to select objects to deploy, you must run the comparison, select the objects, and then save the project.

## Use case sensitive object definition

For databases with case-sensitive collation, enables objects with case-sensitive names to be compared and deployed. 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 or case-sensitive sort order.

Be careful when you change this option. For example, if you create a database snapshot with this option selected and you then compare the snapshot with another database with the option cleared, SQL Compare may produce unexpected errors.

## Force column order

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 deployment. Data will be preserved.

## Do not use transactions in deployment SQL scripts

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

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

## Add WITH ENCRYPTION

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

When SQL Compare creates a snapshot, this option is ignored, and WITH ENCRYPTION isn't saved in the snapshot.

If you use ADD ENCRYPTION on a SQL Server 2005 database, SQL Compare won't be able to display, compare, or deploy the encrypted objects.

## Do not use ALTER ASSEMBLY to change CLR objects

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

If CLR objects are to be deployed, this option forces two rebuilds of the table with conversion to and from strings to update the CLR objects, instead of using ALTER ASSEMBLY. For a detailed explanation, see [Understanding the deployment](#).

This option affects the deployment only.

## 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 Compare considers the next filegroup in the comparison and deployment if the partition scheme is extended. The next filegroup doesn't affect how data is stored.

To ignore next filegroups, clear the check box.

## Disable DDL triggers during deployment

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

DDL triggers can cause problems when you run the deployment. Select this option to disable any enabled DDL triggers before deploying the databases, and re-enable those triggers following deployment.

## Don't include a comment header in the deployment script

When this option is set, the comment header isn't included in the deployment script.

### Add database USE statement

Adds a database USE statement to the top of the deployment script.

This option affects the deployment only.

### 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 Compare decrypts text objects in SQL Server 2008 and SQL Server 2005 databases which were created using the WITH ENCRYPTION option.

When this option isn't selected, text objects in SQL Server 2008 and SQL Server 2005 databases are shown as different, and can't be deployed.

When comparing large databases, selecting this option can result in slower performance.

### Ignore migration scripts for databases

When this option is selected, SQL Compare won't try to retrieve migration scripts when you compare a database.

By default, when you compare a database that has an associated revision number, SQL Compare tries to connect to source control to retrieve any relevant migration scripts.

This option can be useful if you've encountered an error connecting to source control when comparing a database.

### Use migration scripts V2 (beta) instead of V1

Enables the Migrations V2 beta in Compare. V1 migration scripts will be ignored.

Migrations V2 is still a beta feature, so we can't guarantee it will work flawlessly just yet. For more information, see the [SQL Source Control documentation](#).

### Database project compatible script folder output

This option forces script folder output to conform to the style used by the most recent Visual Studio database project type.

### Ignore indexes

Ignores indexes, statistics, unique constraints, and primary keys when comparing and deploying databases.

### Ignore permissions

Ignores permissions on objects when comparing and deploying databases.

### Ignore DML triggers

Ignores DML triggers when comparing and deploying databases.

### Ignore constraint and index names

Ignores the names of indexes, foreign keys, primary keys, and default, unique, and check constraints when comparing databases. The names won't be ignored when the databases are deployed.

## Ignore system named constraint and index names

Ignores the names of system named indexes, statistics, foreign keys, primary keys, and default, unique, and check constraints when comparing and deploying databases.

## Ignore whitespace

Ignores white space (newlines, tabs, spaces, and so on) when comparing databases. White space won't be ignored when the databases are deployed.

## Ignore comments

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

Comments won't be ignored when the objects are deployed.

## Ignore full-text indexing

Ignores full-text catalogs and full-text indexes when comparing and deploying 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 Compare compares and deploys object permissions only for roles, and members of roles that are roles. Users' permissions and role memberships are ignored.

## Ignore statistics

Ignores statistics when comparing and deploying databases.

## Ignore foreign keys

Ignores foreign keys when comparing and deploying databases.

## Ignore check constraints

Ignores check constraints when comparing and deploying databases.

## Ignore identity seed and increment values

For identity properties, ignores only the identity seed and increment values when comparing databases. They won't be ignored when the databases are deployed.

## Ignore fill factor and index padding

Ignores the fill factor and index padding in indexes and primary keys when comparing and deploying databases.

## Ignore INSTEAD OF triggers

Ignores INSTEAD OF DML triggers when comparing and deploying databases.

## Ignore bindings

Ignores bindings on columns and user-defined types when comparing and deploying databases. For example, **sp\_bindrule** and **sp\_bindefault** cl

auses will be ignored.

## Ignore WITH NOCHECK

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

Foreign keys or constraints that are *disabled* aren't ignored.

## Ignore filegroups, partition schemes, and partition functions

Ignores filegroup clauses, partition schemes, and partition functions on tables and keys when comparing and deploying databases. Partition schemes and partition functions aren't displayed in the comparison results.

## Ignore extended properties

Ignores extended properties on objects and databases when comparing and deploying databases.

## Ignore SET QUOTED\_IDENTIFIER and SET ANSI\_NULLS statements

Ignores these SET statements when comparing views, stored procedures, and so on. These statements won't be ignored when the databases are deployed.

## Ignore collations

Ignores collations on character data type columns when comparing and deploying databases.

## Ignore certificates, symmetric keys, and asymmetric keys

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

SQL Server severely restricts access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Compare can't compare all of the properties for a symmetric key.

If certificates, symmetric keys, and asymmetric keys are selected for deployment, only the permissions are deployed.

## Ignore DML 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 comparing and deploying databases. The DDL trigger order isn't affected.

## Ignore event notification on queues

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

Ignores the event notification on queues when comparing and deploying databases.

## Ignore user properties

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

If this option isn't selected, SQL Compare compares user properties, such as the type of user (SQL, Windows, certificate-based, asymmetric key based) and any schema. If a user is selected for deployment, SQL Compare deploys the properties where possible.

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

## Ignore WITH element order



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 comparing and deploying databases.

### Ignore 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 comparing and deploying databases.

### Ignore replication triggers

Ignores replication triggers when comparing and deploying databases.

### Ignore NOT FOR REPLICATION

Ignores the NOT FOR REPLICATION option on foreign keys, identities, check constraints and triggers.

If you select this option, the NOT FOR REPLICATION statement won't be displayed in the object creation script for foreign keys, identities, and check constraints.

In the case of triggers, the NOT FOR REPLICATION statement will be displayed in the object creation script, but will be ignored for the purposes of the comparison. When comparing triggers, you should also select the **Ignore white space** option, but this option will also be applied to all objects in the comparison.

Check constraints and foreign keys that contain the NOT FOR REPLICATION statement in their definition will automatically be flagged as WITH NOCHECK. Use the **Ignore WITH NOCHECK** option to identify these objects as being the same; but this will apply to constraints in all objects.

### Ignore identity property on columns

Ignores the identity property on columns when comparing databases. The identity property won't be ignored when databases are deployed.

### Ignore data compression

This option is used only for SQL Server 2008 databases.

Ignores page and row compression for tables and indexes. When **Ignore filegroups** is selected, compression is automatically ignored for partitioned tables.

### Ignore database name and server name in synonyms

Ignores database names in synonyms when comparing databases.

### Ignore owner authorization on schema objects

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

Ignores authorization clauses on schema-qualified objects when comparing and deploying databases.

### Ignore STATISTICS\_NORECOMPUTE property on indexes

Ignores the STATISTICS\_NORECOMPUTE property on indexes and primary keys.

### Ignore square brackets in object names

Ignores starting and ending square brackets in object names which have been escaped using square brackets. This applies to textual objects such as stored procedures, triggers, etc.

### Ignore tSQLt framework and tests

Ignores the tSQLt schema and its contents, the tSQLtCLR assembly, the SQLCop schema and its contents, and any schemas and their contents with the tSQLt.TestClass extended property set.

### **Ignore encryption of object text (SQL Compare 10.4 and later)**

Ignores WITH ENCRYPTION statements on triggers, views, stored procedures and functions.

This option overrides Add WITH ENCRYPTION.

## Comparing databases on unconnected SQL Servers

You can compare databases on unconnected SQL Server instances by creating SQL Compare snapshots or scripts folders from the databases.

### Using snapshots

To compare databases on unconnected SQL servers using snapshots:

1. Create a snapshot of the database(s).  
For more information, see: [Working with snapshots](#).
2. Copy the snapshot to the required location.
3. Do one of the following:
  - [Create a project](#) that compares the snapshot with a database
  - [Create a project](#) that compares the snapshot with another snapshot, or with a scripts folder.
4. Run the comparison on the project.
5. [Select the database objects](#) you want to include in the deployment.
6. Use the [deployment wizard](#) to generate the deployment script.

When a snapshot is the target, the deployment creates a script to update the database from which the snapshot was created. Snapshots cannot be modified directly.

### Using scripts folders

To compare databases on unconnected SQL servers using scripts folders:

1. Create a scripts folder from the database(s).  
For more information, see: [Working with scripts folders](#).
2. Copy the scripts folder to the required location.
3. Do one of the following:
  - [Create a project](#) that compares the scripts folder with a database
  - [Create a project](#) that compares the scripts folder with another set of scripts
4. Run the comparison on the project.
5. [Select the database objects](#) that you want to include in the deployment.
6. Use the [deployment wizard](#) to update the database schema, object script files or generate a deployment script.

Scripts folders are only available in SQL Compare professional edition.

## Permissions required to use SQL Compare

The permissions required to compare and deploy a database using SQL Compare depend on the objects in your database schema and the version of SQL Server you are using.

For example, a user belonging to the `PUBLIC` role can compare a SQL Server 2000 database. However, they can not compare encrypted stored procedures without `dbo` permissions.

To perform a deployment, we recommend you have `dbo` permissions.

If you create a deployment script to run later, you are also recommended to run the script as a `dbo` user. This ensures that any objects created will have the correct schema, permissions, and authorizations.

- If you have insufficient permissions, some objects may be missing from the comparison results. For example, User Defined Types do not appear in the comparison results if you do not have permissions for the schema they belong to.
- A deployment script generated from incomplete comparison results may fail or produce unexpected results. For example, the schema may refer to a User Defined Type that does not exist.
- If you do not have `dbo` rights, granting the `VIEW DEFINITION` permission is sufficient to compare unencrypted objects in SQL Server 2005 or later databases. However, `sysadmin` permissions are required to decrypt encrypted stored procedures.
- If you're using SQL Server 2008 or later, we recommend you have `SELECT` permission for the system views `sys.sql_expression_dependencies`. You may experience poor performance when comparing databases if you don't have this permission.
- If you're using SQL Server 2008 or later, you may require `VIEW SERVER STATE` permissions to compare some encrypted objects.
- When deploying, a user must have permission to make all of the modifications listed in the **Summary** tab at the end of the deployment wizard, or the deployment may fail. If a deployment fails, in most circumstances changes are rolled back. SQL Compare uses *transactions* to do this. However, there are some circumstances in which this is not possible.

For more general information on permissions, see your SQL Server documentation.

# Reviewing the comparison results

These pages describe ways of dealing with the comparison results.

- [Viewing the comparison results](#)
- [Viewing the SQL differences](#)
- [Using filters](#)
- [Comparison warnings](#)
- [Understanding the comparison results](#)
- [Generating a report](#)

## Viewing the comparison results

When you have compared the data sources, SQL Compare displays the comparison results in the upper (Results) pane. The upper pane displays all the objects you can select for deployment.

To compare the data sources again using the same project configuration, and update the comparison results, click



### Refresh.

You can define which objects are displayed by setting up a [filter](#). You can also view the differences in the objects' creation scripts in the [SQL Differences](#) pane.

Objects are displayed in groups, either by how they differ between the two data sources, by the object type, or by whether they match what is typed in the **Find** box. When you first run the comparison, the objects are grouped by *Type of difference*.

## Object groups

To view the objects in a group, click



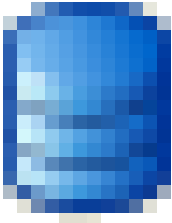
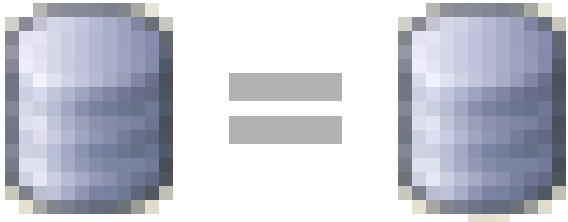
, or click the group heading.

The screenshot shows the Results pane in SQL Compare. At the top, it compares '(LOCAL) WidgetStaging' and '(LOCAL) WidgetProduction'. Below this, it indicates 'Objects excluded by filter: 0' and '2 of 5 objects selected for synchronization'. The main table has columns for 'Type', 'Object Name', and 'Object Name'. It is grouped into four categories: '4 objects that exist in both but are different' (containing WidgetPrices, WidgetReferences, Widgets, and CurrentPrices), '1 object that exists only in (local).WidgetStaging' (CurrentPrices), and '10 identical objects'. The 'Type' column uses icons to indicate the difference: a red 'X' for objects that exist in both but are different, and a blue cylinder for objects that exist only in the source.

Type	Object Name	Object Name
4 objects that exist in both but are different 2 of 4		
Table	WidgetPrices	WidgetPrices
Table	WidgetReferences	WidgetReferences
Table	Widgets	Widgets
View	CurrentPrices	CurrentPrices
1 object that exists only in (local).WidgetStaging 0 of 1		
10 identical objects =		

When objects are grouped by *Type of difference*, the **Type** column indicates the difference:

	objects that exist in both data sources but are different
	objects that exist only in the <i>Source</i>


	<p>objects that exist only in the <i>Target</i></p>
	<p>objects that exist in both data sources and are identical</p>

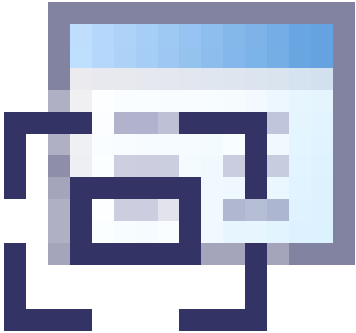


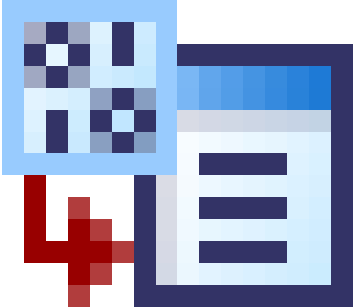

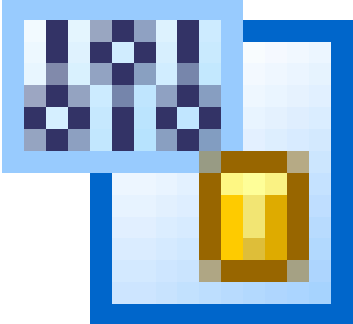

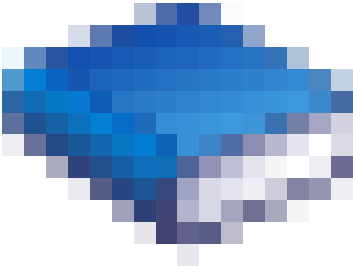
You can also group by object type. In the **Group by** box, select *Type of object*.

Type	Owner	Object Name	Object Name
94 tables			84 of 94
22 views			3 of 22
9 stored procedures			0 of 9
f(x) 12 functions			12 of 12
10 roles			=
6 user defined types			0 of 6
1 DDL trigger			1 of 1
18 schemas			5 of 5 (18)
6 XML schema collections			6 of 6

When objects are grouped by *Type of object*, the **Type** column indicates the type of object.


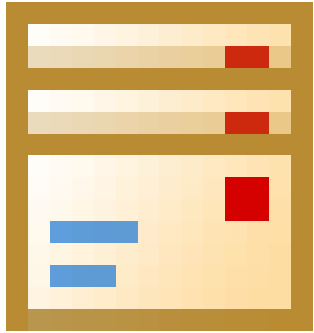
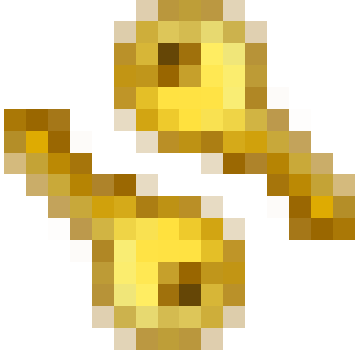
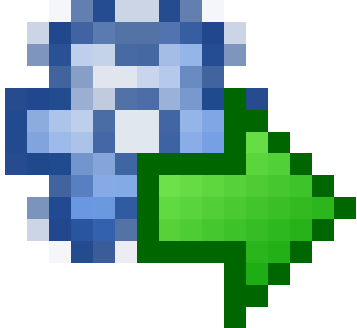

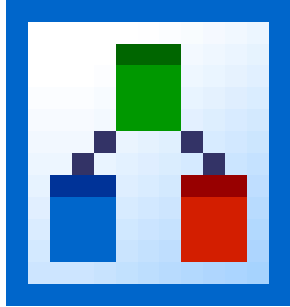
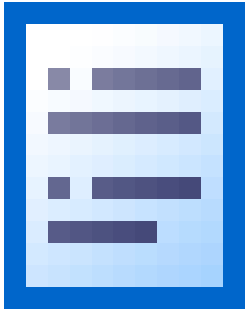
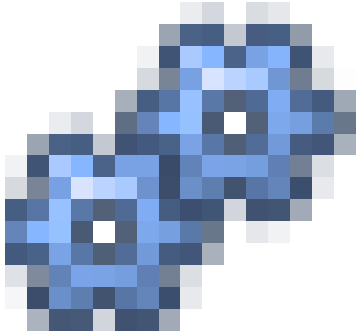
For SQL Server 2000 and later:

	<p>Tables</p>		<p>Roles</p>
---	---------------	--	--------------

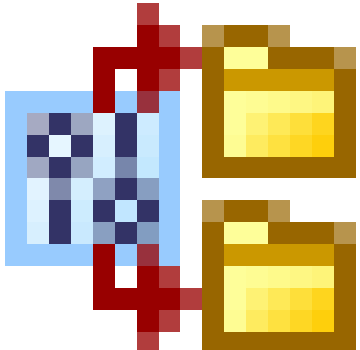

	Views		Rules
	Stored Procedures		Defaults
	Functions		User Defined Types
	Users		Full Text Catalogs

For SQL Server 2008 and SQL Server 2005:



	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)
---	-------------------	--	---

The **Owner** column displays the owner (or schema) of the object. This column is shown only when there is more than one owner (or schema) in the database.

## Objects covered by migration scripts

Objects that are covered by migration scripts are indicated in the comparison results with an icon:

Object Name /	<input type="checkbox"/>	Object Name
	<input type="checkbox"/>	0 of 1
vAssocSeqLineItems	<input checked="" type="checkbox"/>	vAssocSeqLineItems
=		
AdventureWorksDWBldVersion	=	AdventureWorksDWBldVersion
DatabaseLog	=	DatabaseLog

For more information, see [Deploying with migration scripts](#).

You can view the migration script that covers the object by right-clicking the object in the comparison results, and then clicking **View Migration Script**.

## Selecting objects for deployment

The grouping bars show the number of objects in a group that will be deployed.

Use the check boxes in the middle of the upper pane to include an object in the deployment or exclude it from the deployment. If only some objects in a group are selected for deployment, the check box on the grouping bar enters a mixed state:

Type	Owner	Object Name /	<input type="checkbox"/>	Object Name
6 views				
			<input checked="" type="checkbox"/>	2 of 6
View	HumanResources	vEmployee	<input type="checkbox"/>	Some objects in this group selected
View	HumanResources	vEmployeeDepar...	<input checked="" type="checkbox"/>	
View	HumanResources	vEmployeeDepar...	<input checked="" type="checkbox"/>	
View	HumanResources	vJobCandidate	<input type="checkbox"/>	
View	HumanResources	vJobCandidateE...	<input type="checkbox"/>	
View	HumanResources	vJobCandidateE...	<input type="checkbox"/>	
3 stored procedures				
			<input checked="" type="checkbox"/>	3 of 3
Stored Procedure	HumanResources	uspUpdateEmplo...	<input checked="" type="checkbox"/>	All objects in this group selected
Stored Procedure	HumanResources	uspUpdateEmplo...	<input checked="" type="checkbox"/>	
Stored Procedure	HumanResources	uspUpdateEmplo...	<input checked="" type="checkbox"/>	

The check box for an object grouping affects all objects in that grouping. So, if you save a project with a grouping set to include all objects in that group, the next time you use this project, all objects in that group are selected for deployment, even if they did not exist when the project was last used.

## Finding an object

To locate objects, type the search text in the **Find** box. To select a recent search, click the **Find** arrow button ▼

As you type, objects are grouped in the upper pane by whether they match or do not match what you type:

Type	Object Name	<input type="checkbox"/>	Object Name
3 objects match 'widget'			0 of 3
Table	WidgetPrices	<input type="checkbox"/>	WidgetPrices
Table	WidgetReferences	<input type="checkbox"/>	WidgetReferences
Table	Widgets	<input type="checkbox"/>	Widgets
12 objects do not match 'widget'			0 of 2 (12)

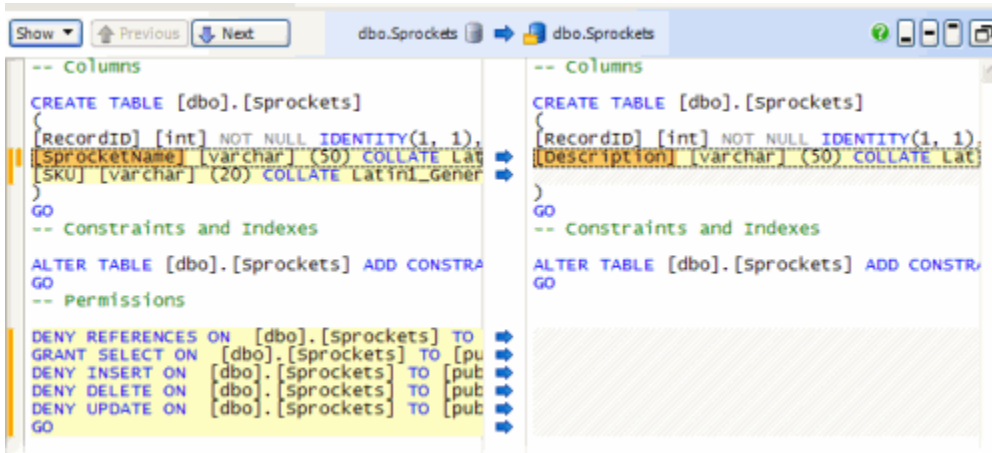
SQL Compare searches object names and owners (schemas).

To clear the **Find** box click the **X** button.

The search is not case-sensitive.

## Viewing the SQL differences

The lower (SQL Differences) pane displays a side-by-side listing of differences in the creation SQL script for an object. To display the SQL Differences pane, click an object in the upper (Results) pane.



You can adjust the height of the SQL Differences by dragging it up or down, or by using the



buttons to move the pane to a fixed height.

Click

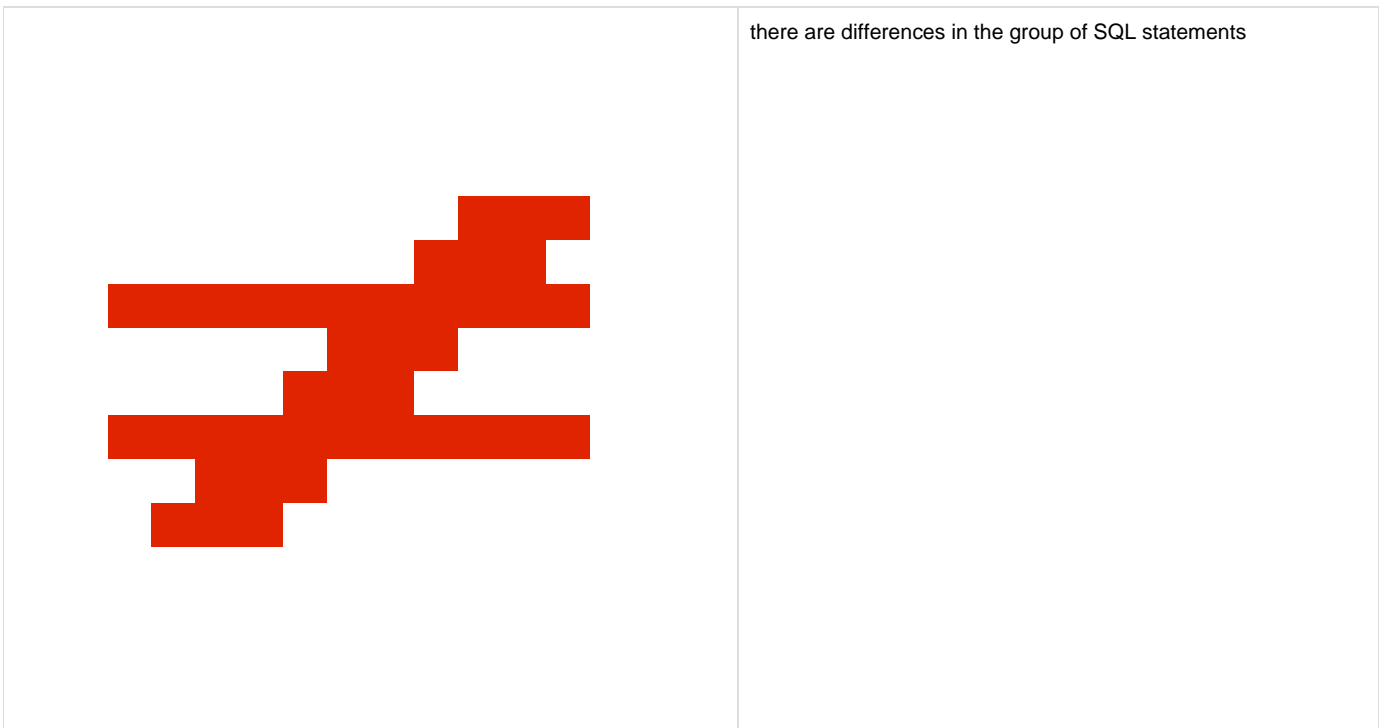


to open the SQL Differences pane in a separate window.

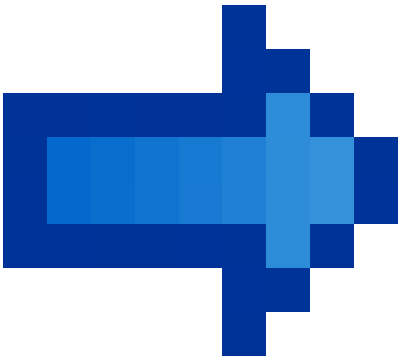
## Viewing differences

Lines that contain differences are displayed with a shaded background; text within a line that is different is displayed with a darker shaded background.

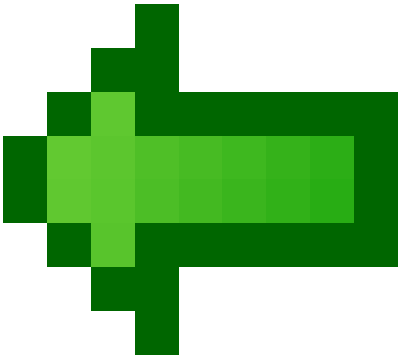
The central column shows the type of difference:



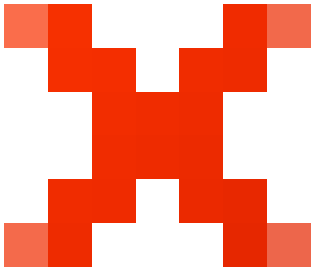
this line of SQL will be added if you deploy this object (the direction of the arrow indicates the database that will be updated)



this line of SQL will be added if you deploy this object (the direction of the arrow indicates the database that will be updated)



this line of SQL will be removed if you deploy this object



There is no icon if the two lines are identical.

Use



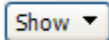
Next and



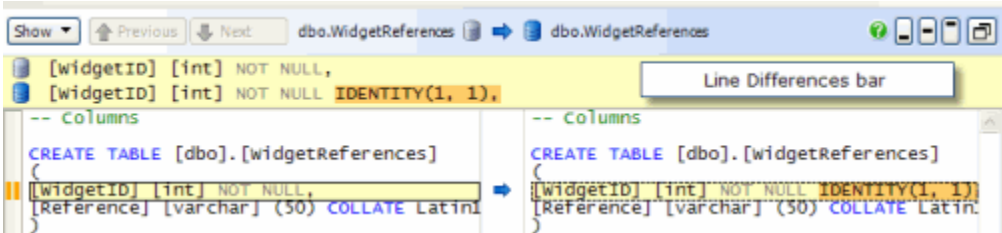
Previous to go to lines that contain a difference. You can also use Alt + Up Arrow and Alt + Down Arrow.

### The Line Differences bar

Click



and select **Line Differences** to display the Line Differences bar. The two versions of the line are shown one on top of the other. This is especially useful when the lines are too long to view all the text:



### Scroll bar

Colored blocks in the left scroll bar indicate the location in the script of lines that are different.



A line of SQL that exists only in the object script on the left



A line of SQL that exists only in the object script on the right



A line of SQL that exists in both object scripts, but is different

## Searching and copying the creation script

To search the SQL statements, right-click and select **Find**.

To search the SQL statements in the other data source, click **Find on this side**.

The search isn't case-sensitive.

## Viewing the object deployment script

To view the deployment script for the selected object, right-click in the SQL Differences pane, and then click **Show Object Deployment Script**.

The **Single Object Deployment Script** dialog box is displayed.

You are recommended to deploy only using the full deployment script created by the deployment wizard, because the single objects script cannot account for dependencies. If you use the object deployment script to deploy a single object, there may be unexpected results, or the deployment may fail.

To view the full deployment script that will make the two databases identical, use the [deployment wizard](#).



## Using filters

You can use filters to define which objects are displayed in the upper (Results) pane. For example, you may be interested in only tables, or in only some of the stored procedures owned by a specific schema. You can set up a filter to show you only the objects matching these conditions.

You can set up a filter for object types, and edit the *filter rules* to restrict which specific objects are included in or excluded from the upper pane.

The default filter is *Nothing Excluded*. If you edit this filter, *Custom\** is shown in the **Filter** box. An asterisk is displayed next to the name of any filter you edit, to show that there are unsaved changes.

You can save your filters for use in multiple projects. To choose a different filter, in the **Filter** box select the filter you want to use; if the filter is not listed, select *Open filter from file* and browse to the filter.

## Filter rules

You can define which object types the filter includes in the comparison results using the object type check boxes in the **Filter** pane.

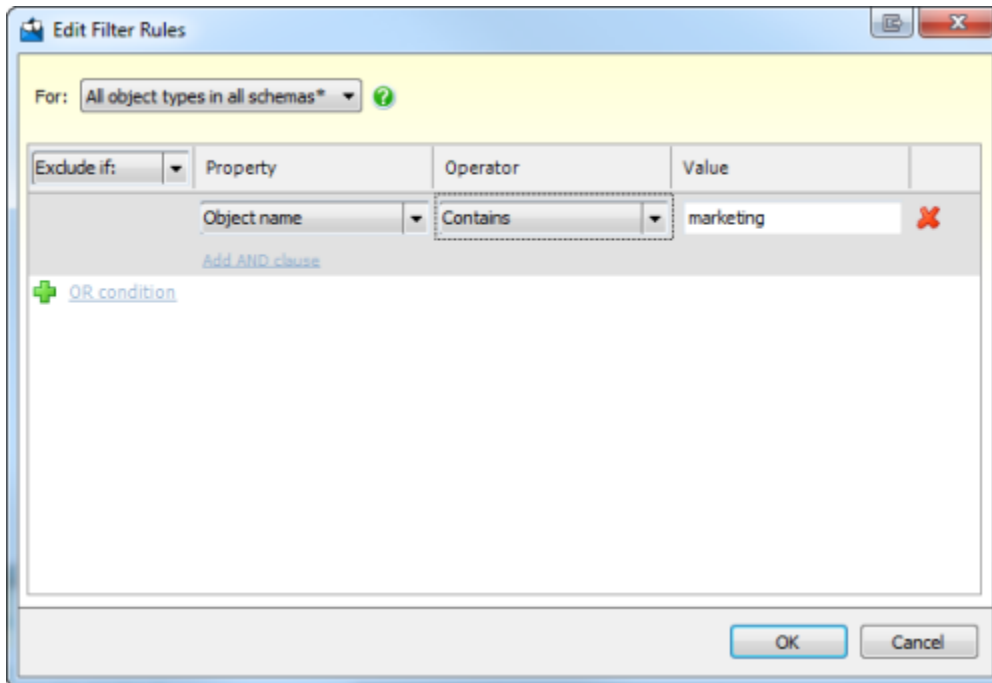
You can also create rules to control the specific objects that a filter includes or excludes. To edit the filter rules:

On the **Filter** pane, click



**Edit Filter Rules.**

The **Edit Filter Rules** dialog box is displayed:



You can select individual object types or *All object types in all schemas*, and edit the rules applying to those objects.

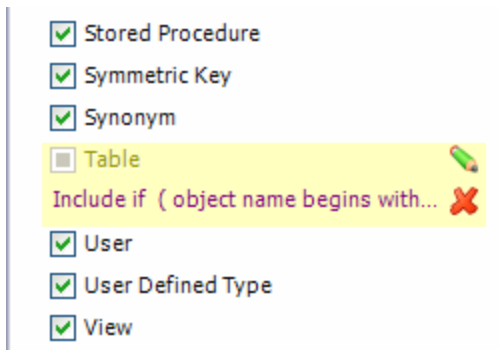
In SQL Compare 10.2 and later, \_ (underscore) works as a wildcard matching any character, as it does in SQL. This means that a filter that excludes objects beginning with \_ will exclude all objects.


When you create a filter rule, its conditions are displayed under the name of any object types it applies to.

To edit the filter rule for an object type, move the mouse pointer over it and click the



(Edit) button that becomes available:



To clear the filter rule for an object type, move the mouse pointer over it and click the  (Clear) button that becomes available.

You can clear all current filter rules by selecting the filter *Nothing Excluded* from the **Filter** box.

## Saving and deleting filters

Saved filters are available in subsequent projects, and can be selected from the **Filter** box. Saved filters have the file extension *.scpf*

- To save the current filter, on the **Filter** pane, click



### Save

In the **Save Filter** dialog box, type the name for the filter.

When you save an edited filter, you can either save it with the same name to overwrite it, or change the name to create a new filter.

- To delete the current filter, on the **Filter** pane, click



### Delete

You will be prompted to confirm. When you delete the current filter, *Custom\** is shown in the **Filter** box, and the conditions for the filter you just deleted remain set until you select another filter or make any changes.

Any unsaved changes to a filter are lost when you select another filter.

## Copying or moving filters

Filters are saved in your *%USERPROFILE%\Documents\SQL Compare\Filters* folder. All files in this folder with the *.scpf* extension will be displayed in the *Filter* drop-down list.

To migrate filters to another computer, copy them from the *%USERPROFILE%\Documents\SQL Compare\Filters* folder to the corresponding folder on that computer.

You must copy filters to the folder of the user who will be using SQL Compare on that computer.

## Using filters from the SQL Compare command line

To use a filter from the command line, use the */filter* switch and specify the filter you want to use, including the *.scpf* extension. For example:

```
SQLCompare.exe /s1:server1 /db1:widgetlive /s2:server1 /db2:widgetdev  
/filter:myfilter.scpf
```

You can use this switch to call any filter that you have saved in SQL Compare. It will also look for filters in the current working directory of your command prompt. To use a filter that you have set up on another machine, copy or move the filter as described in '*Copying or moving filters*', above.

You can also specify a filter by absolute or relative path. For example, using the working directory `C:\SQLCompare`, you can call the filter `myfilter.scpf` from `C:\SQLCompare\filters\myfilter.scpf` with a relative path:

```
C:\SQLCompare>SQLCompare.exe /s1:server1 /db1:widgetlive /s2:server1 /db2:widgetdev
/filter:filters\myfilter.scpf
```

Or using an absolute path:

```
SQLCompare.exe /s1:server1 /db1:widgetlive /s2:server1 /db2:widgetdev
/filter:C:\SQLCompare\filters\myfilter.scpf
```

## Filters and Find

A filter defines which objects are displayed in the upper (Results) pane; using **Find** changes how those objects are grouped. If an object matches the term in the **Find** box *and* is excluded by a filter, it is not displayed in the upper pane.

**Find** groups results by whether they match or do not match the term typed in the **Find** box. Objects not matching the **Find** term are still displayed in the upper pane, but this group is shown as collapsed by default.

Only objects displayed in the upper pane can be selected for deployment.

### Example: excluding objects by type


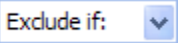
If you want to view comparison results for only Tables and Views:

1. On the **Filter** pane, clear the *All object types included* check box.  
This ensures that no objects are displayed in the upper (Results) pane.
2. Select the *View* and *Table* check boxes.

The filter is applied. Only tables and views are displayed in the upper pane.

### Example: excluding objects from a specific schema


If you want to view comparison results that exclude any objects owned by the schema *Marketing*:

1. On the **Filter** pane, click the  **Edit Filter Rules** button.  
The **Edit Filter Rules** dialog box is displayed.
2. In the **For** box, ensure that *All object types in all schemas* is selected.
3. In the  box, select *Exclude if*.
4. Under **Property**, select *Schema name*.
5. Under **Operator**, select *Equals*.
6. Under **Value**, type *Marketing*.
7. Click **OK**.

The filter is applied. No objects owned by the schema *Marketing* are displayed in the upper pane.

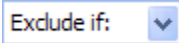
### Example: including only those tables with a specific name or a particular owner

If you want to view comparison results for all the tables that have names beginning with *Marketing*, as well as all those owned by the schema *Marketing*, regardless of their names:

1. On the **Filter** pane, clear the *All object types included* check box.  
This ensures that no objects are displayed in the upper pane.
2. Click the  button for *Table*.

The **Edit Filter Rule** dialog box is displayed.

3. In the



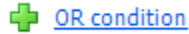
box, select *Include if*.

4. Under **Property**, select *Object name*.

5. Under **Operator**, select *Begins with*.

6. Under **Value**, type *Marketing*.

7. Click



A new OR condition becomes available.

8. Under **Property**, select *Schema name*.

9. Under **Operator**, select *Equals*.

10. Under **Value**, type *Marketing*.

11. Click **OK**.

The filter is applied. Only objects owned by the schema *Marketing*, or with names that begin with *Marketing* are displayed in the upper pane.

## Filters and deployment

The current filter affects which objects you can select for deployment:

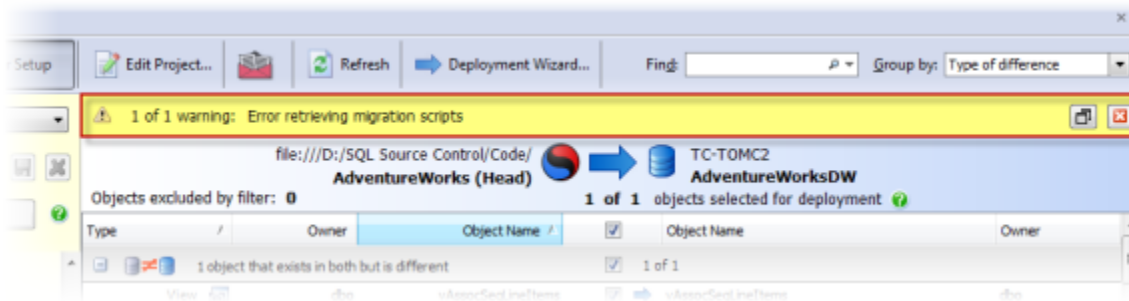
- When you use the filter to exclude an object or object type, it is removed from the upper pane and cannot be selected for deployment.
- If you select an object for deployment and later set up a filter that excludes it, the object is not deployed.

Objects excluded by the filter are only included in the deployment as dependencies. If an excluded object is referenced by an object you selected for deployment, you will be notified of this dependency, and can choose to deploy the affected object on the **Review Dependencies** page of the deployment wizard.

## Comparison warnings

In the comparison results, SQL Compare can display warnings about the comparison.

Warnings are displayed in a bar at the top of the comparison results:



To view the warning messages in more detail, expand the warning bar by clicking



. If you close the warning bar, you'll need to refresh the comparison to view the warnings again.

The most common warnings that SQL Compare may display are summarized below.

### Error retrieving migration scripts

This warning is displayed if there has been an error connecting to source control and [migration scripts](#) couldn't be retrieved.

If you don't want SQL Compare to try and retrieve migration scripts for future comparisons with a project, edit the project and select the **Ignore migration scripts for databases** [project option](#).

### Migration scripts detected - SQL Compare Professional required

This warning is displayed if SQL Compare has detected [migration scripts](#) that apply to the comparison, but it can't retrieve them; SQL Compare Professional edition is required.

To use migration scripts with SQL Compare, you must upgrade to SQL Compare Professional edition.

## Understanding the comparison results

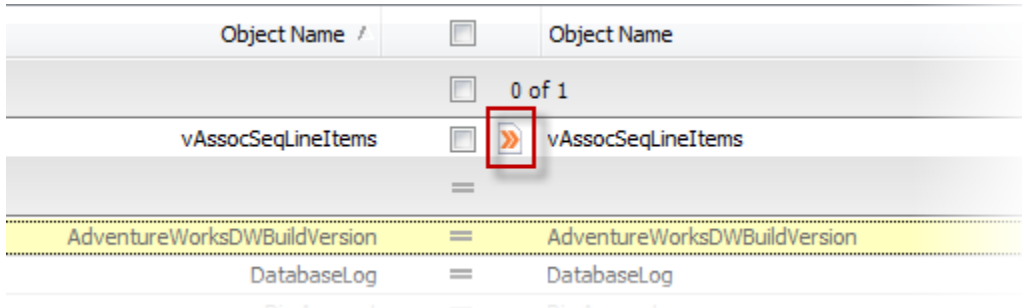
### Objects with different owners or schemas

Objects that are the same but have different owners or schemas 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.


If you want to compare objects that are the same but have different owners, you can [remap the owners](#) when you set up your comparison project.

### Objects covered by migration scripts

Objects that are covered by migration scripts are indicated in the comparison results with an icon:



The screenshot shows a comparison table with two columns labeled 'Object Name'. The table lists several objects: 'vAssocSeqLineItems', 'AdventureWorksDWBUILDVersion', and 'DatabaseLog'. Each object has a checkbox and an icon to its left. The icon for 'vAssocSeqLineItems' is a red square with a white arrow pointing right, indicating it is covered by a migration script. Above this icon, the text '0 of 1' is displayed. The row for 'AdventureWorksDWBUILDVersion' is highlighted in yellow. Below the table, there are two buttons: 'View Migration Script' and 'View Differences'.

Object Name		Object Name
	<input type="checkbox"/>	
	<input type="checkbox"/>	0 of 1
vAssocSeqLineItems	<input type="checkbox"/> 	vAssocSeqLineItems
	=	
AdventureWorksDWBUILDVersion	=	AdventureWorksDWBUILDVersion
DatabaseLog	=	DatabaseLog

For more information, see: [Deploying with migration scripts](#).

You can view the migration script that covers the object by right-clicking the object in the comparison results, and then clicking **View Migration Script**.

### Database-level permissions

For SQL Server 2000, differences in database-level permissions are not detected by SQL Compare. 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 comparison project, it is recommended that you use roles.

For SQL Server 2008 and SQL Server 2005, differences in database-level permissions are detected.

### Object-level permissions

When you connect to a database for which your user is not the database owner, SQL Compare displays only those objects for which you have been granted permissions. In addition, the SQL Differences pane will display only the object-level permissions for the authenticated user; other users' permissions will not be displayed.

The deployment script may fail if the authenticated user does not have full object-level permissions.

If a permission is given using WITH GRANT OPTION (for example **GRANT SELECT ON <objectname> TO <username> WITH GRANT OPTION**) it will not be recognized. Permissions assigned that do not use WITH GRANT OPTION are detected.

### Extended stored procedures

SQL Compare does not compare extended stored procedures.

### Encrypted database objects

If you are comparing a SQL Server 2000 database that contains an encrypted user-defined function, stored procedure, trigger, or view and you have system administrator permissions, SQL Compare decrypts the object and you can view its internal SQL in the SQL Differences pane of the main window.

In SQL Compare version 7.1 (and later) you have the option to decrypt text objects in SQL Server 2008 and SQL Server 2005 databases created using the WITH ENCRYPTION option.

Disabling this option can result in faster performance. To disable this option, on the **Options** tab of the Project Configuration dialog box, clear the **Decrypt encrypted objects on 2005 and 2008 databases** check box.

When this option is disabled, SQL Compare can't compare the encrypted objects, or display their creation SQL scripts, and can't deploy them.

SQL Compare version 7.0 (and earlier) can't decrypt objects that are encrypted in a SQL Server 2005 or SQL Server 2008 database. If an encrypted object which can't be decrypted exists in both databases, it is shown under the **objects that exist in both but are different** group in the comparison results in the main window (select *Type of Difference* in the **Group by** box to arrange objects by difference). SQL Compare can't compare the encrypted objects, or display their creation SQL scripts, and can't deploy them.

## Certificates, symmetric keys, and asymmetric keys

SQL Server severely restricts access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Compare can't compare all of the properties for a symmetric key.

To ignore all certificates, symmetric keys, and asymmetric keys in the deployment, on the Project Configuration dialog box, select the [project option \*Ignore certificates, symmetric and asymmetric keys\*](#).

## Replication

SQL Compare does not compare stored procedures that have been created for replication.

## Generating a report

When you have run the comparison, you can export the results as a report. You can choose to include the objects [selected for deployment](#), all objects with differences, and, optionally, identical objects.

To generate a report, on the **Tools** menu, click **Generate Comparison Results Report**.


The report groups objects according to whether they will be updated, created or dropped, or are identical. Selecting *Type of object* or *N o groups* in the **Group by** box in the main window is not reflected in the report.

You can create a report in one of four formats:

## Interactive HTML

You can expand and collapse the object groups in the Interactive HTML report. By clicking an object you can view the differences in the object creation scripts.

**SQL Compare Report: 13 Feb 2009 12:14 PM**

SQL2008 ( live )  SQL2008 ( live )  
**SprocketStaging** **SprocketProduction**

Type	Owner	Object Name		Object Name	Owner
<b>Different</b>					
4					
Table	dbo	SprocketPrices	≠	SprocketPrices	dbo
Table	dbo	SprocketReferences	≠	SprocketReferences	dbo
Table	dbo	SprocketParts	≠	SprocketParts	dbo
View	dbo	CurrentPrices	≠	CurrentPrices	dbo
<b>Create</b>					
1					
Stored Procedure	dbo	prActivatePrices	→		
<b>Drop</b>					
0					
<b>Identical</b>					
23					

SQL View

<pre>-- Columns CREATE TABLE [dbo].[SprocketPrices] ( [RecordID] [int] NOT NULL IDENTITY(1, 1), [SprocketID] [int] NULL, [Price] [money] NULL, [DateValidFrom] [datetime] NULL CONSTRAINT DF_SprocketPrices_DateValidFrom DEFAULT (getdate()), [DateValidTo] [datetime] NULL</pre>	≠	<pre>-- Columns CREATE TABLE [dbo].[SprocketPrices] ( [RecordID] [int] NOT NULL IDENTITY(1, 1), [SprocketID] [int] NULL, [Price] [money] NULL</pre>
--	---	---

## Simple HTML

The **view SQL** links in the Simple HTML report enable you to view the differences in the creation script for an object. The **Summary** links enable you to locate an object in the object groupings.



SQL Compare Report:  
**SQL2008.SprocketStaging ( live ) vs SQL2008.SprocketProduction ( live )**  
 13 Feb 2009 12:18 PM

**Different**

Type	Owner	Object	Object	Owner
Table	dbo	SprocketPrices	<> SprocketPrices	dbo <a href="#">view SQL</a>
Table	dbo	SprocketReferences	<> SprocketReferences	dbo <a href="#">view SQL</a>
Table	dbo	Sprockets	<> Sprockets	dbo <a href="#">view SQL</a>
View	dbo	CurrentPrices	<> CurrentPrices	dbo <a href="#">view SQL</a>

**In SQL2008.SprocketStaging only**

Type	Owner	Object	Object	Owner
Stored Procedure	dbo	prcActivatePrices	->	<a href="#">view SQL</a>

**In SQL2008.SprocketProduction only**

Type	Owner	Object	Object	Owner
------	-------	--------	--------	-------

**Identical**

Type	Owner	Object	Object	Owner
Role	dbo	db_accessadmin	= db_accessadmin	dbo <a href="#">view SQL</a>
Role	dbo	db_backupoperator	= db_backupoperator	dbo <a href="#">view SQL</a>
Role	dbo	db_datareader	= db_datareader	dbo <a href="#">view SQL</a>
Role	dbo	db_datawriter	= db_datawriter	dbo <a href="#">view SQL</a>
Role	dbo	db_ddadmin	= db_ddadmin	dbo <a href="#">view SQL</a>
Role	dbo	db_denydatareader	= db_denydatareader	dbo <a href="#">view SQL</a>
Role	dbo	db_denydatawriter	= db_denydatawriter	dbo <a href="#">view SQL</a>
Role	dbo	db_owner	= db_owner	dbo <a href="#">view SQL</a>
Role	dbo	db_securityadmin	= db_securityadmin	dbo <a href="#">view SQL</a>
Role	dbo	public	= public	dbo <a href="#">view SQL</a>

**SQL Scripts**

Type	Owner	Object	Object	Owner
Table	dbo	SprocketPrices	SprocketPrices	dbo <a href="#">Summary</a>
-- Columns				
		<>	-- Columns	
CREATE TABLE [dbo].[SprocketPrices]			CREATE TABLE [dbo].[SprocketPrices]	
(			(	
[RecordID] [int] NOT NULL IDENTITY(1, 1),			[RecordID] [int] NOT NULL IDENTITY(1, 1),	
[SprocketID] [int] NULL,			[SprocketID] [int] NULL,	
[Price] [money] NULL,		<>	[Price] [money] NULL	
[DateValidFrom] [datetime] NULL CONSTRAINT		<>		
[DF_SprocketPrices_DateValidFrom] DEFAULT		<>		
(getdate()),		<>		
[DateValidTo] [datetime] NULL,		<>		
[Active] [char] (1) COLLATE Latin1_General_CI_AS NULL		<>		
CONSTRAINT [DF_SprocketPrices_Active] DEFAULT ('N')		<>		
)			)	
GO			GO	

## Excel

An Excel report does not show object creation scripts or differences.

## XML

The XML report is an XML file listing the objects you selected, their creation scripts, and whether they are different, identical, or present in only one of the data sources. This is useful, for example, if you want to apply your own formatting using XSLT.

# Deploying data sources

When you have reviewed the [comparison results](#), you can run the deployment wizard to create the SQL script to deploy the selected objects:

1. Select the objects that you want to deploy.
2. Click



**Deployment Wizard** to start the deployment wizard.

## Selecting the objects to deploy

By default, no objects are selected for deployment. You can modify the selection by using the check boxes in the middle of the upper (Results) pane:

Type	Owner	Object Name	<input type="checkbox"/>	Object Name
4 objects that exist in both but are different				3 of 4
Table	dbo	WidgetPrices	<input checked="" type="checkbox"/>	WidgetPrices
Table	dbo	WidgetReferences	<input checked="" type="checkbox"/>	WidgetReferences
Table	dbo	Widgets	<input type="checkbox"/>	Widgets
View	dbo	CurrentPrices	<input checked="" type="checkbox"/>	CurrentPrices
1 object that exists only in TC-TOMC2.WidgetStaging				1 of 1
Stored Procedure	dbo	prcActivatePrices	<input checked="" type="checkbox"/>	
23 identical objects				=

Object selections are remembered when you save a project.

To select all the objects in a group for deployment, select the check box on the grouping bar for that group; alternatively, you can right-click within the group and click **Include All Objects in this Group**.

To exclude all objects displayed in a group from the deployment, clear the check box; or, you can right-click the group and click **Exclude All Objects in this Group**.

The top level check box in the central column enables you to include or exclude all objects that are displayed in the upper (Results) pane:

Type	Owner	Object Name	<input checked="" type="checkbox"/>	Object Name
4 objects that exist in both but are different				4 of 4
Table	dbo	WidgetPrices	<input checked="" type="checkbox"/>	WidgetPrices
Table	dbo	WidgetReferences	<input checked="" type="checkbox"/>	WidgetReferences
Table	dbo	Widgets	<input checked="" type="checkbox"/>	Widgets
View	dbo	CurrentPrices	<input checked="" type="checkbox"/>	CurrentPrices
1 object that exists only in TC-TOMC2.WidgetStaging				1 of 1
Stored Procedure	dbo	prcActivatePrices	<input checked="" type="checkbox"/>	
23 identical objects				=

You can [filter](#) the object types and use the **Find** box to assist you with your selection.

If you select an object for deployment and then set up a filter that excludes that object type, the object is not included in the deployment. Only selected objects that are shown in the comparison results are included.

For example, if you select *TableA* for deployment, and then set up a filter to exclude all tables, no tables are shown. If you subsequently remove the filter, *TableA* is displayed and remains selected for deployment.

### Example: Selecting objects that exist only in the source

1. In the **Filter** pane, to the left of the results, ensure that the default filter *Nothing Excluded* is selected, so that all the objects are displayed in the upper (Results) pane.  
No objects are now excluded by the filter.
2. Ensure the top level check box in the central column is cleared, or on the **Actions** menu select **Exclude All**.  
No objects are now selected for deployment.
3. From the **Group by** box, select *Type of difference* if it is not already selected. On the grouping bar for **objects that exist only in <source database name>**, select the check box to include all of the objects in this group.

### Example: Selecting only stored procedures

1. From the **Group by** box, select *Type of difference* if it is not already selected.  
The objects are now grouped by whether they are different, identical, or present in only one of the data sources.
2. In the **Filter** pane, select and clear the *All object types included* check box, and then select the check box for *Stored Procedures*.  
Only Stored Procedures are now displayed in the upper pane.
3. Select the top level check box in the central column to include all objects in the deployment.

### Example: Selecting only tables that exist in both databases but are different

1. From the **Group by** box, select *Type of difference* if it is not already selected.  
The objects are now grouped by whether they are different, identical, or present in only one of the data sources.
2. In the **Filter** pane, clear the *All object types included* check box, and then select the check box for *Tables*.  
Only Tables are now displayed in the upper pane.
3. Clear the top level check box in the central column to exclude all objects from the deployment.  
No objects are now selected for deployment.
4. On the grouping bar for **objects that exist in both but are different**, select the check box to include all of the tables in this group in the deployment.

## Backing up before deployment

The **Configure Backup** page of the deployment wizard allows you to perform a full backup using either [Red Gate SQL Backup](#), or SQL Server native backups.

### Backup name and location

For **SQL Server native**:

1. Type the file path in the **Backup folder** box or click



to specify the path using the folder browser. By default, **Backup folder** is set to the default backup folder for the SQL Server instance.

2. Type the file name in the box to the right of the **Backup folder** box.

For **Red Gate SQL Backup**:

1. Type the file path in the **Backup folder** box or click



to specify the path using the folder browser. By default, **Backup folder** is set to the folder specified in the SQL Backup options for the SQL Server instance. If no backup file locations have been set up, SQL Backup uses the SQL Server instance's default backup folder.

2. Specify the file name in the box to the right of the **Backup folder** box. By default, the file name is set to `<AUTO>.sqb`; SQL Backup uses the SQL Backup options to generate the backup file path and file name. If no backup file locations have been set up, SQL Backup uses the SQL Server instance's default format for file names.

To change the file name, clear the **Name file automatically** check box, and type the required file name. You can use SQL Backup tags, if required. For information about tags, see *File Location Tags* in the SQL Backup online help.

To specify a network path in the **Backup folder** box, type the full path, including the server name, for example `\\ServerName\MyFolder`

The file path is relative to the selected SQL Server. For example, if you have chosen to back up a database on a remote SQL Server instance called ServerA and you specify a local path such as `C:\Backups`, the backup files will be created on the C: drive on ServerA, not on the local computer.

Select the **Overwrite existing backup files of the same name** check box if you want to overwrite any files of the same name that exist for the file path you specified in the **Backup Folder** box.

If a file of the same name exists already and you have not chosen to overwrite it, the backup will fail if you are using SQL Backup.

### Backup compression (Red Gate SQL Backup only)

If you are using SQL Backup to back up the target database, you can choose from the three compression levels described below. Generally, the smaller the resulting backup file, the slower the backup process.

To compress the backup, select the **Compress backup** check box and select the compression level by moving the slider.

- **Compression level 3**  
Compression level 3 uses the zlib compression algorithm. This compression level generates the smallest backup files in most cases, but it uses the most CPU cycles and takes the longest to complete.
- **Compression level 2**  
This compression level uses the zlib compression algorithm, and is a variation of compression level 3. On average, the backup process is 15% to 25% faster than when compression level 3 is used, and 12% to 14% fewer CPU cycles are used. Backup files are usually 4% to 6% larger.
- **Compression level 1**  
This is the default compression level. It is the fastest compression, but results in larger backup files. On average, the backup process is 10% to 20% faster than when compression level 2 is used, and 20% to 33% fewer CPU cycles are used. Backup files are usually 5% to 9% larger than those produced by compression level 2. However, if a database contains frequently repeated values, compression level 1 can produce backup files that are smaller than if you used compression level 2 or 3. For example, this may occur for a database that contains the results of Microsoft SQL Profiler trace sessions.

If SQL Backup Lite is installed on the SQL Server, you can choose only compression level 1.

### Backup encryption (Red Gate SQL Backup only)

If you are using SQL Backup to back up the target database, you can encrypt the backup by selecting the **Encrypt backup** check box, then typing a password for the backup in **Password**, and again in **Confirm**.

If SQL Backup Pro is installed on the SQL Server, you can choose 128-bit or 256-bit encryption; if SQL Backup Standard is installed on the SQL Server, you can choose only 128-bit encryption; if SQL Backup Lite is installed on the SQL Server, you cannot encrypt the backup.

**You must remember your password**; if you do not, you will not be able to access the encrypted backup.

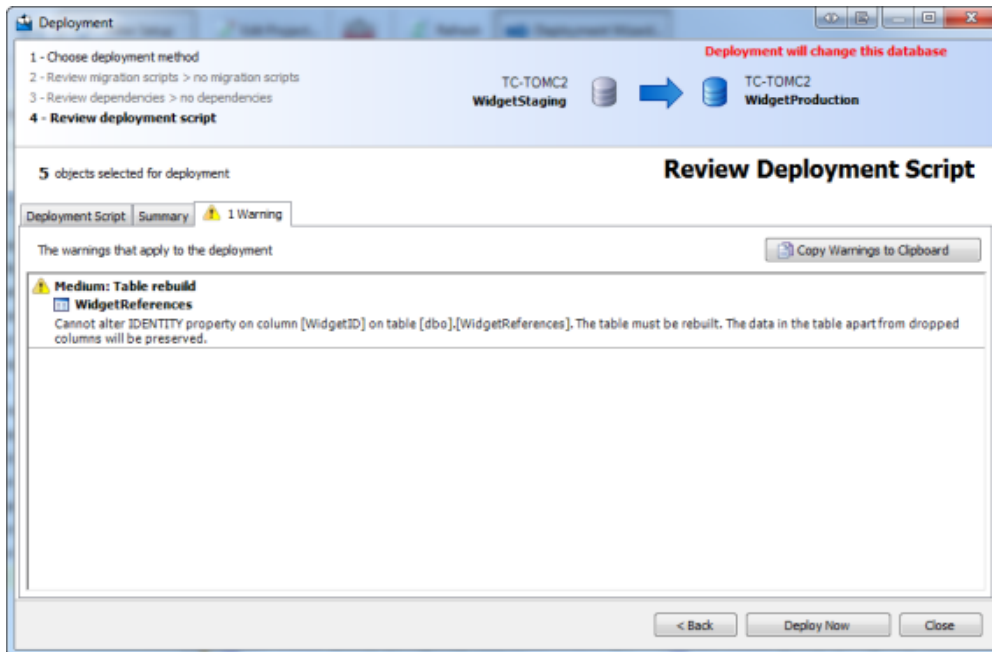
### Using multiple threads (Red Gate SQL Backup only)

If you are using SQL Backup to back up the target database and you are using a multi-processor system, using multiple threads can speed up the backup process. Select the **Use multiple threads** check box and type or select the number of threads up to a maximum of 32. You are recommended to start with one thread fewer than the number of processors. For example, if you are using four processors, start with three threads.

For details of how you can find out the most effective number of threads to use for your setup, see [Optimizing Backup Speed](#) in the SQL Backup help.

## Deployment warnings

When the deployment wizard displays the **Review** page, you can click the **Warnings** tab to view any warnings about inefficiencies in the script, or reasons the script may fail.



The most common warnings that SQL Compare may display are summarized below. SQL Compare may also display warnings specific to SQL Server 2008 and SQL Server 2005 databases.

### Table rebuild

To rebuild a table, SQL Compare drops the table and recreates it. SQL Compare creates a temporary table to store data so that data is not lost when the table is dropped.

SQL Compare rebuilds a table when:

- a table's filegroup has changed  
You can ignore filegroups by selecting the **Ignore filegroups, partition schemes, and partition functions** project option. By default, filegroups are ignored.
- a column cannot be altered  
For example, this warning is displayed if a column is to be changed from data type *text* to *varchar*.
- a property of a column cannot be altered  
For example, this warning is displayed if a column is an identity in one database but not in the other.
- the identity column on a table has changed  
For example, this warning is displayed if the seed has changed.
- the column order on a table has changed  
This warning is displayed only if you select the **Force column order** in your **project options** and the order of columns in a table has changed. To ignore column order, clear the project option.
- column x on table y must be added but has no default and does not allow NULL values  
If the table contains data, the deployment script will not work. To avoid this, add a default to the column in the source database, or set it to allow NULL values.
- column x on table y must be added and does not allow NULL values; the default z must be bound to it  
A table in the source database contains an additional column that is set to NOT NULL, and has a default set.

### Non-standard filegroups

You must create the filegroups manually before you perform the deployment. SQL Compare lists the filegroups that must be created.

- You can ignore filegroups by selecting the **Ignore filegroups, partition schemes, and partition functions** project option. By default, filegroups are ignored.

### Column will be truncated

For example, SQL Compare displays this warning when the length that is defined for the column has changed (such as *varchar(50)* to *varchar(30)*)

This may result in loss of data.

### Invalid cast

For example, SQL Compare displays this warning when:

- a data type has changed from *text* to *varchar*
- a user-defined data type has changed

In these cases, the deployment may fail.

### Loss of precision or data

For example, SQL Compare displays this warning when the precision or scale of a decimal column has changed.

There may be loss of precision or data, or the deployment may fail.

### Default password

If a user or application role must be created, SQL Compare displays details of the user or application role, and the default password that will be applied. The default password is **p@ssw0rd**.

### Corrupt login

SQL Compare displays this warning when the login for a user is not defined. For example, when a database is restored, the association between users and logins is not preserved. To link a user to a login, you use **sp\_change\_users\_login**. For more information, refer to your SQL documentation.

### Statistics creation

Statistics will be created with default settings. You may need to modify the statistics manually.

### No default value for column

The default value that is set in the source database will be applied where appropriate. If there is no default value, the update may fail.

### Full-text information is being added to the database

The target database may not be full-text enabled (for example, because it has recently been restored from a backup). Ensure any full-text catalogs have been rebuilt on the target database before you run the script.

You can ignore full-text indexing by selecting **Ignore full text indexing** in your [project options](#).

## Understanding the deployment

This topic provides information that may help you to understand the behavior of the deployment script.

### Column order

Column order isn't forced unless you select the **Force column order** project option.

For example, your source database has a table that contains *ColA* and *ColB*, in that order, and your target database has the same table but with *ColB* then *ColA*. If **Force column order** isn't selected, SQL Compare shows the tables as identical objects in the comparison results. If the option is selected, SQL Compare shows the columns as different objects; you can select the objects for deployment.

When column order is to be changed in a database, SQL Compare provides a **warning** in the **Summary** page of the deployment wizard to notify you that the table will be rebuilt. SQL Compare uses temporary tables to make sure that any data in the table isn't lost.

### Renamed columns

SQL Compare 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 Compare renames the column as appropriate.

However, if the names and data types are very different, SQL Compare may consider the renamed column to be a completely different column. In this case, if *ColA* in your source database is renamed to *ColB* in your target database, when SQL Compare creates the deployment script, *ColA* will be created in the target database as a new column and *ColB* will be deleted. To avoid data loss, before you deploy the databases you must take care to preserve any data in the two columns, and merge them following the deployment.

### Renamed objects

SQL Compare will detect inconsistencies in SQL Server when the name of an object such as a stored procedure, view, or function has been changed using *sp\_rename*. In SQL Server, using *sp\_rename* doesn't change the corresponding name in the object definition. SQL Compare will fix this inconsistency if the object needs to be altered by editing the name within the object definition to match the object name.

It isn't considered best practise to use *sp\_rename* to rename stored procedures, triggers, user-defined functions, or views.

### Updated views

If your views have not been updated by the deployment script and they contain a `SELECT *` statement, you must refresh them 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's not best practice to use `SELECT *` statements in views; we recommend you specify an explicit column list.

### Database diagrams

SQL Compare doesn't compare or deploy database diagrams.

### System objects

SQL Compare doesn't compare or deploy system objects, except for users, roles, and system schemas.

### Replication

If objects that are used in replication are deployed, errors may occur. For example, SQL Compare can't 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*. SQL Compare references only the user name, so that *Computer1\User1* and *Computer2\User1* would be considered as the same. Therefore, if you intend to deploy users, make sure that their user names are different.



SQL Compare compares and deploys changes to users, such as changes to permissions. However, SQL Compare doesn't compare or deploy modifications to user passwords.

## Filegroups

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

When a filegroup is to be changed in a database, SQL Compare provides a [warning](#) in the **Deployment script** page of the deployment wizard to notify you that the table will be rebuilt. SQL Compare uses temporary tables to make sure that any data in the table isn't lost.

## Encrypted database objects

If you are deploying a SQL Server 2000 database that contains an encrypted user-defined function, stored procedure, trigger, or view and you have system administrator permissions, SQL Compare decrypts the object and you can view its internal SQL in the deployment script. If you do not have system administrator privileges, you can't deploy the encrypted object.

In SQL Compare version 7.1 (and later) you can decrypt text objects in SQL Server 2005 and SQL Server 2008 databases created using the WITH ENCRYPTION option.

Disabling this option can result in faster performance. To disable this option, on the **Options** tab of the Project Configuration dialog box, clear the **Decrypt encrypted objects on 2005 and 2008 databases** check box.

When this option is disabled, SQL Compare can't compare the encrypted objects, or display their creation SQL, and can't deploy them.

SQL Compare version 7.0 (and earlier) can't decrypt objects that are encrypted in a SQL Server 2005 or SQL Server 2008 database. If an encrypted object that can't be decrypted exists in both databases, it is shown under the **objects that exist in both but are different** group in the comparison results in the main window (select *Type of Difference* in the **Group by** box to arrange objects by difference). SQL Compare can't compare the encrypted objects, or display their creation SQL, and can't deploy them.

## CLR assemblies

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

If SQL Compare determines that it would not be possible to use ALTER ASSEMBLY, any table that contains a CLR type from the updated assembly 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 deployment script may fail, or the data may be corrupted.

To force SQL Compare to use the double table-rebuild method, select the **Do not use ALTER ASSEMBLY to change CLR objects** [project option](#).

## Partition schemes and partition 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 Compare ignores filegroups. However, if you clear the [project option](#) **Ignore filegroups, partition schemes, and partition functions**, SQL Compare deploys 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 deployment also requires a table to be rebuilt twice, the CLR assembly and the partition scheme are deployed at the same time.

## Certificates, symmetric keys, and asymmetric keys

SQL Server severely restricts access to certificates, symmetric keys, and asymmetric keys. Consequently, SQL Compare can't compare all of the properties for a symmetric key.

If certificates, symmetric keys, and asymmetric keys are selected for deployment, only the permissions are deployed.

To ignore all certificates, symmetric keys, and asymmetric keys in the deployment, select the **Ignore certificates, symmetric and asymmetric keys** project option.

## Extended properties on databases

Extended properties on databases that differ are not displayed in the comparison results, but are always deployed. If you do not want them to be deployed, select the **Ignore extended properties** project option.

## Numbered stored procedures

SQL Compare doesn't deploy numbered stored procedures. However, you can deploy them by running the deployment script in your SQL application.

## Using the Deployment Wizard

When you have [selected the objects](#) that you want to include in the deployment, use the deployment wizard to create the deployment script. To open the deployment wizard, click



### Deployment Wizard.

There are five possible steps in the wizard:

1. [Choose deployment method](#)  
Create and save a script, perform the deployment from SQL Compare, or update a scripts folder.
2. [Configure backup](#) (optional step)  
If, on the first page of the wizard, you selected the option to back up before deploying, this additional step is shown. If you do not choose to back up, this step is not part of the wizard.
3. [Review migrations scripts](#)  
View a list of migrations scripts you can select to include in the deployment.
4. [Review dependencies](#)  
View a list of any objects that reference or are referenced by those you have chosen to deploy. You can choose to deploy these objects.
5. [Review deployment script](#)  
View the deployment script, review a summary of the deployment actions, and see information about any warnings.

### Deploying backups

When you have selected a backup as the target, the deployment wizard creates a script to update the database from which the backup was created. Backups cannot be modified directly.

When a backup is the source, and a database is the target, the deployment script will deploy the database from the backup.

For more information, see: [Working with backups](#).

### Deploying snapshots

When a SQL Compare snapshot is the target, deployment creates a script to update the database from which the snapshot was created. Snapshots cannot be modified directly.

When a snapshot is the source, and a database is the target, the deployment script will deploy the database from the snapshot.

This is useful, for example, if you want to [roll back changes](#). If you have made changes to a database, and created a snapshot before deployment, you can then set the snapshot as the source, and the database as the target, to roll back the changes.

For more information, see: [Working with snapshots](#).

### Deploying scripts folders

When a scripts folder is the target, you can either:

- Create a deployment script to update the database from which the scripts folder was created, or
- Update files in the scripts folder directly

When an object is dropped during deployment, its script file is not deleted.

If a scripts folder is the target, and any of the script files that will be modified are designated as read-only, a warning is displayed. If you click **Yes** to continue, these files will be made writable so that they can be modified. This may occur, for example, when you are working with a source control system that sets files to read-only status in some situations.

For more information, see: [Working with scripts folders](#).

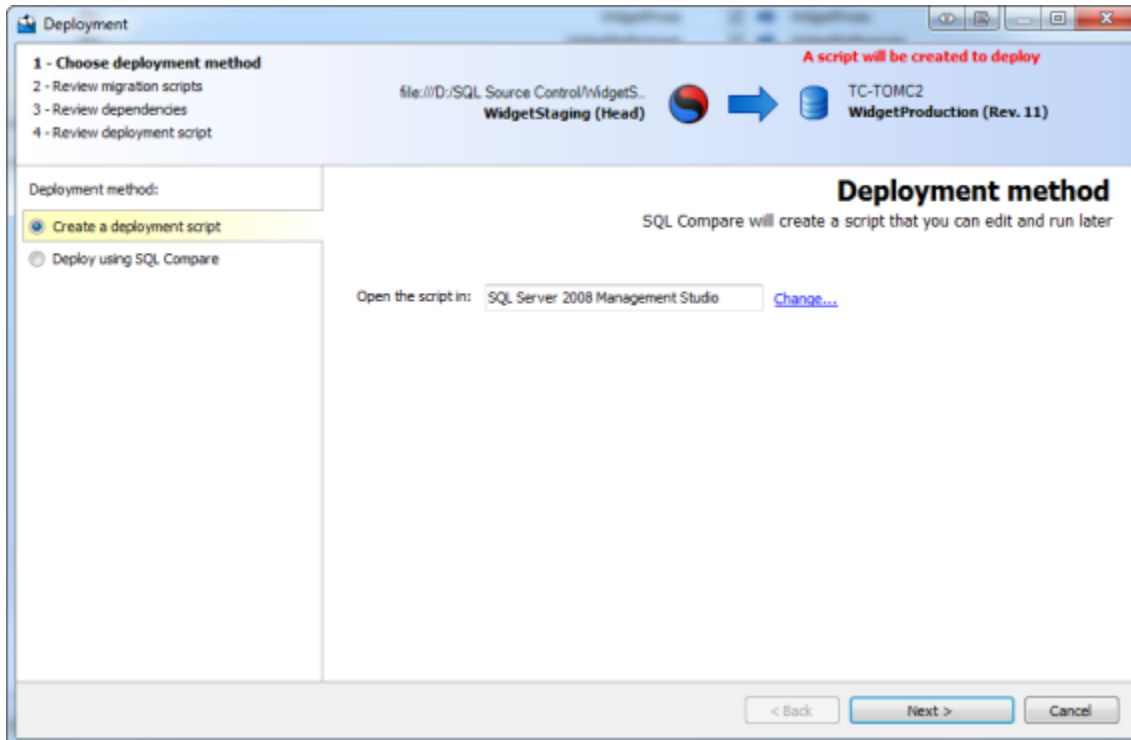
### Additional information

If SQL Compare is unable to deploy the data sources, an error message is displayed, and where possible all changes are rolled back. However, if you have selected the [project option "Do not use transactions in deployment SQL scripts"](#), the changes are not rolled back.

## 1. Choose deployment method

On the first page of the deployment wizard you can choose to create and save a deployment script, perform the deployment using SQL Compare, or update a scripts folder:

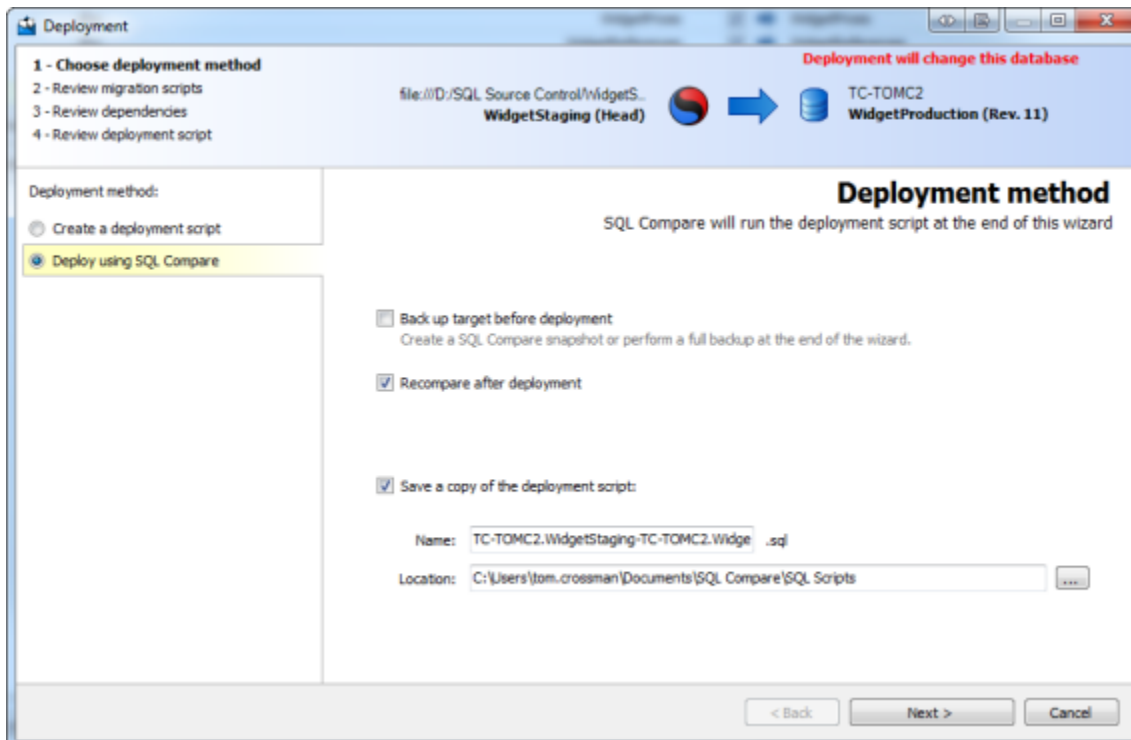
## Create a deployment script



If you choose to create a deployment script, on the **Review** page you can save a copy of the script or open it in your SQL editor.

To change the application you use to open the script, click **Change**. The Application Options dialog box is displayed, and you can specify the default application used to edit SQL Scripts.

## Deploy using SQL Compare



If you choose to deploy using SQL Compare, on the **Review** page you can save a copy of the script and exit the wizard, or click **Deploy Now** to perform the deployment.

If the target is a scripts folder, this option is instead replaced by **Update the scripts folder**, and the SQL script files in the target are modified when you deploy.

If you select **Back up target before deployment**, a step is added to the deployment wizard allowing you to specify details of the backup.

If the target is a backup or a snapshot, the option to deploy using SQL Compare is not available. Instead, deployment creates a script to modify the data source form which the snapshot or backup was created.

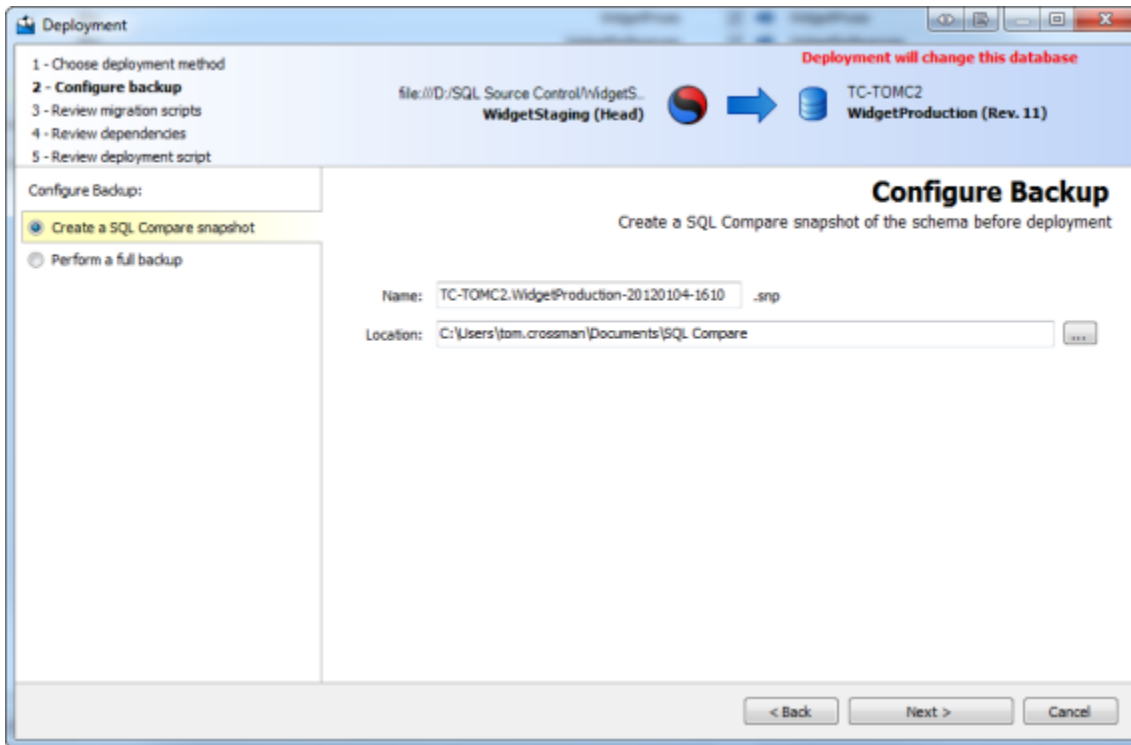
## 2. Configure backup (optional step)

If you selected the option to back up before deployment, the **Configure backup** page is added to the deployment wizard.

This page allows you to create a SQL Compare schema snapshot, or a full backup. You can use either Red Gate SQL Backup, or native SQL Server to perform the backup.

If the target is a scripts folder, you cannot perform a full backup before deployment. You can only create a snapshot from a scripts folder.

### **Create a SQL Compare snapshot**

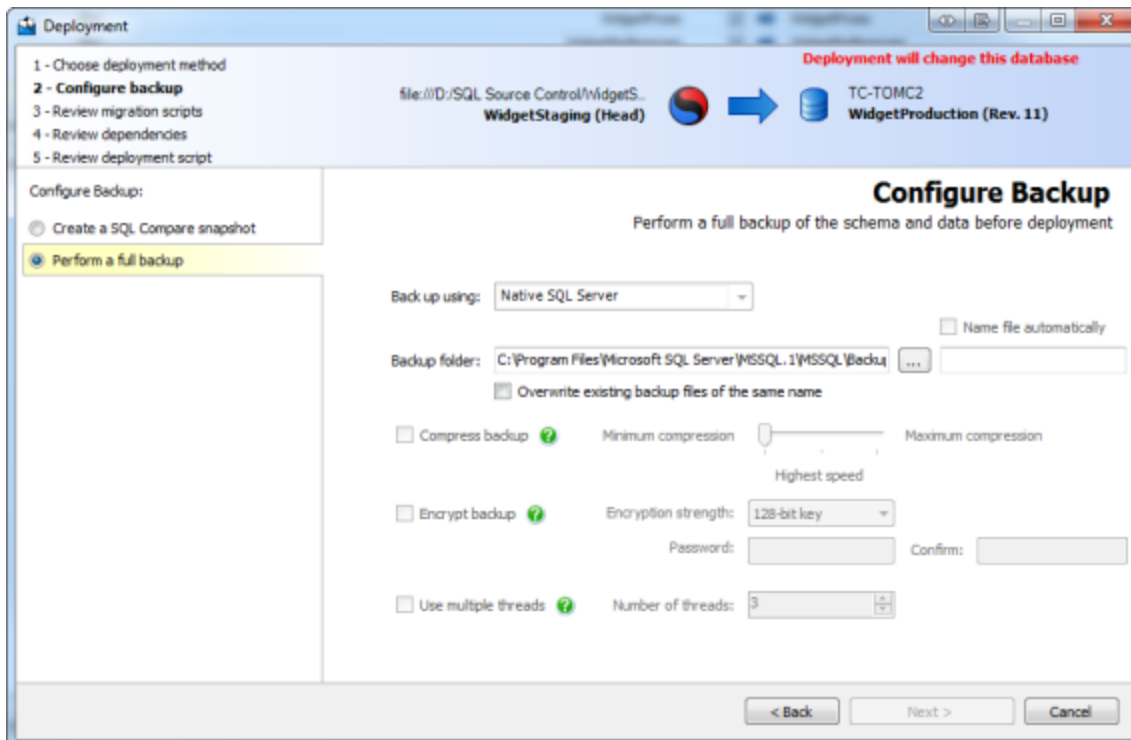


A SQL Compare schema snapshot is a binary file containing information about the structure of a database; it does not contain any table data.

This is useful, for example, if you want to roll back changes.

For more information, see: [Working with snapshots](#).

### **Perform a full backup**



When the target is a database, you can perform a full backup. In the **Back up using** box , select:

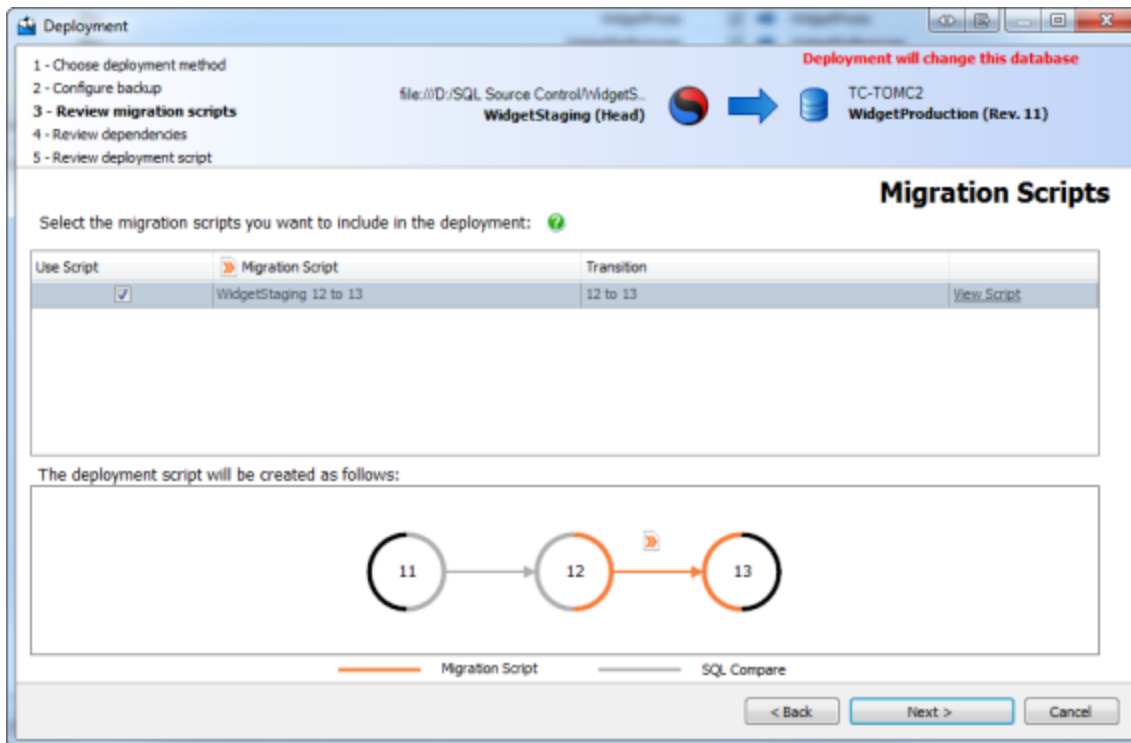
- *Native SQL Server* to create a backup using the native SQL Server BACKUP command
- *Red Gate SQL Backup* to create a backup using SQL Backup version 4 or later

To use Red Gate SQL Backup, you must have the SQL Backup server components installed on the SQL Server instance of the target database.

For more information, see: [Backing up before deployment.](#)

### 3. Review migrations scripts

If SQL Compare detects any migrations scripts that apply to the deployment, the **Review migrations scripts** page of the wizard is displayed:



Migrations scripts are customizable change scripts that are committed to source control and re-used in deployment. For more information, see [Working with migration scripts](#) in the SQL Source Control support center.

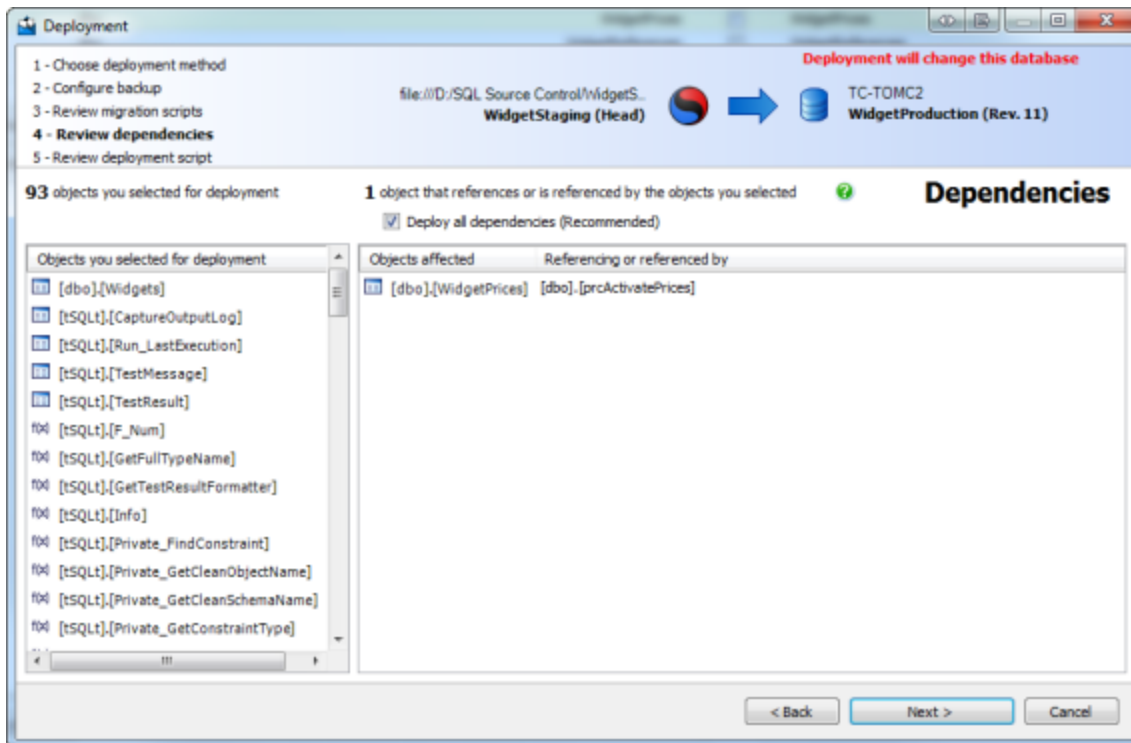
You can select the migrations scripts you want to include in the deployment using the check boxes.

### **Selecting overlapped migrations scripts**

By default, SQL Compare selects a migration script with a transition that covers the most versions. You must deselect this default selection before you can select any overlapped migrations scripts.

## **4. Review dependencies**

When you run the deployment wizard, SQL Compare checks for dependencies:



All the objects you selected for deployment are listed. If they reference or are referenced by objects you did not select, and these dependencies affect the deployment, they are listed on this page.

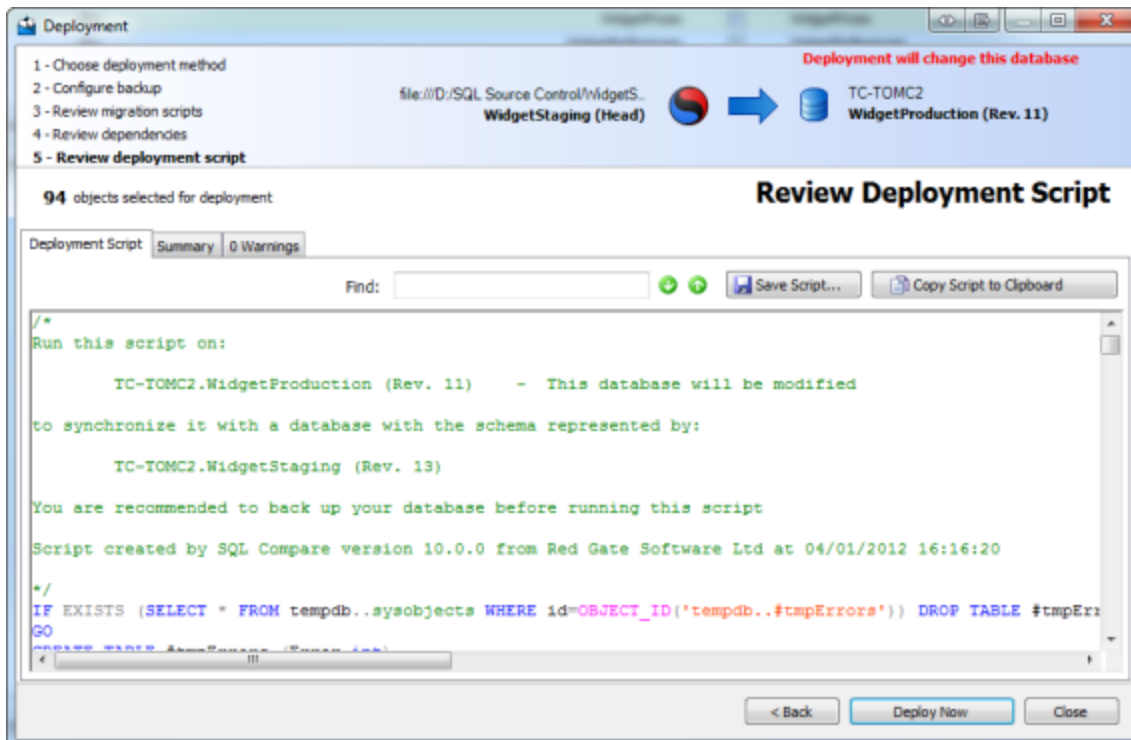
For example, if a stored procedure references a table, and you do not include that table in the deployment, the table is listed here.

By default, SQL Compare includes these objects in the deployment. To exclude them, clear the **Deploy all dependencies** check box.

We recommend you deploy all objects with dependencies. If you don't, objects may be left in an invalid state, or the deployment may fail.

## 5. Review deployment script





There are three tabs on the Review page:

- **Deployment** script shows the script to deploy the data sources. You can search the script, save it, or copy it to the clipboard.
- **Summary** shows a synopsis of the actions in the deployment script. You can view the summary grouped by the objects affected, by the type of modification, or by the order in which the script modifies the target.
- **Warnings** shows a list of any warnings about unexpected behavior that may occur when you deploy the databases. For more information, see: [Deployment warnings](#).

If you are updating or creating a scripts folder, the **Files** tab lists the object creation and data script files modified or created during deployment.

When you have reviewed the script, click **Deploy Now** to perform the deployment.

## SQL Compare may drop and create CLR assemblies

When SQL Compare modifies a CLR assembly in a database as part of deployment, it will run an ALTER ASSEMBLY query whenever possible. However, there are some circumstances where an assembly must be dropped and re-created.

SQL Compare will drop an assembly and create a new one in the following circumstances:

- The assembly depends on another assembly that needs to be recreated
- The assembly metadata is different between the assembly in either database
- The assembly metadata is invalid or otherwise could not be compared
- Either assembly contains no files
- Either assembly is less than 96 bytes in size

## Rollback on script failure or cancellation

If a deployment script fails, or if it is cancelled, in most circumstances changes are rolled back. SQL Compare uses *transactions* to do this. However, there are some circumstances when this isn't possible:

- if full-text information must be altered (for example: within a transaction, catalogs can't be dropped, and indexing can't be added to a column)
- if users and roles need to be created, altered, or deleted (for example: within a transaction, a user can't be created, or added to a role)

In these cases, SQL Compare rolls back all the changes it can. Your database will be in an undetermined state and you should run another comparison.

If you've selected the project option **Do not use transactions in deployment SQL scripts** to remove transactions from the deployment SQL script, no changes are rolled back when the script fails or is cancelled. This may be useful if you want to run a script up to the point of failure; for example, for debugging.

SQL Compare always warns you if it can't roll back changes.

## Working with other data sources

SQL Compare can compare backup files (both native .bak files and SQL Backup .sqb files), SQL Compare snapshots, and scripts folders.

- [Working with backups](#)
- [Working with snapshots](#)
- [Working with scripts folders](#)
- [Using a DACPAC as a data source](#)

## Working with backups

SQL Compare enables you to compare a backup with other data sources. This is useful, for example, when you want to retrieve the schema from a backup and compare it with your database without running a restore operation or copying the backup from a remote network.

If you are comparing two backups, you do not need SQL Server to be installed on your computer.

- Comparing backups is available only in SQL Compare Professional edition.
- SQL Compare can retrieve the schema from full or differential backups. However, it does not support partial, filegroup, or transaction log backups.
- When you run a comparison using a backup, SQL Compare locks the backup files when it reads them, and you can't overwrite, move, or delete them.
- SQL Compare does not read the log records of backup files, so if the database schema was modified while the backup was being created, it may not be shown as modified in the comparison results.
- You can specify a backup as the target; however, note that backups can't be modified.

## Comparing and deploying backups

You can:

- compare a backup with another data source  
For more information, see: [Setting data sources](#).
- create a deployment script from a backup

When you have selected a backup as the target, the deployment wizard creates a script to update the database from which the backup was created. Backups can't be modified directly.

When a backup is the source, and a database is the target, the deployment script will deploy the database from the backup.

For more information, see: [Using the Deployment Wizard](#).

## Compatibility with backups

You can compare backups from SQL Server 2008, SQL Server 2005 and SQL Server 2000 databases. SQL Compare supports:

- native SQL Server backups  
You can use backups created with SQL Server native compression (including row-level, page-level and file-level compression).

SQL Server 2008 encrypted backups are not supported.

- SQL Backup backups  
You can use backups created with SQL Backup version 3 or later; you can use compressed or encrypted backups.

When you set up a comparison that uses a backup, SQL Compare does not support tables that do not have a primary key, unique index, or unique constraint.

## Working with snapshots

SQL Compare enables you to create a snapshot of a database and compare it with another data source. A SQL Compare snapshot is a binary file containing information about the structure of a database; it does not contain any table data.

You can use snapshots:

- for simple version control of databases
- to compare databases on unconnected SQL servers

## Creating snapshots

To create a new SQL Compare snapshot, on the **File** menu, select **Create Snapshot**. Alternatively, on the Project Configuration dialog box, select a snapshot as one of the data sources, and click **Create Snapshot**.

The **Create New Snapshot** dialog box is displayed:

**Create New Snapshot**

**Create snapshot**

**Data source details**

Data source type: Database

Server: TC-TOMC2

Windows authentication

SQL Server authentication

User name: sa

Password: \*\*\*\*\*

Database: AdventureWorks

Revision: 5

**Snapshot creation properties**

Snapshot name: AdventureWorks .sn timer

Create in: C:\Users\tom.crossman\Documents\SQL Compare\

Auto detect case sensitivity

Treat items as case sensitive

Decrypt encrypted objects on 2005 and 2008 databases

Create Snapshot Cancel

Under **Data source details**, specify the details of the source for the snapshot. You can create a snapshot from a database, backup, scripts folder, or another snapshot.

Under **Snapshot creation properties**, specify a name and location for the snapshot.

SQL Compare automatically detects the case sensitivity of the data source. If you want to override this, clear the **Auto detect case sensitivity** checkbox, and select the **Treat items as case sensitive** checkbox if required.

By default, SQL Compare decrypts text objects in SQL Server 2008 and SQL Server 2005 databases that were created using the WITH ENCRYPTION option. When comparing large databases, selecting this option can result in slower performance.

- When the source is a **snapshot**, the case sensitivity and decryption options are not available
- When the source is a **scripts folder**, the decryption option is not available

You can also create a snapshot using Red Gate Snapper, a free tool that creates SQL Compare snapshots from SQL Server databases. You can download Red Gate Snapper from the [free Red Gate tools page](#).

## Comparing and deploying snapshots

You can:

- compare a snapshot with another data source  
See: [Setting data sources](#).
- create a deployment script from a snapshot

When a SQL Compare snapshot is the target, deployment creates a script to update the database from which the snapshot was created. Snapshots cannot be modified directly.

When a snapshot is the source, and a database is the target, the deployment script will deploy the database with the snapshot.

This is useful, for example, if you want to [roll back changes](#). If you have made changes to a database, and created a snapshot before deployment, you can then set the snapshot as the source, and the database as the target, to roll back the changes.

For more information, see: [Using the Deployment Wizard](#).

## Compatibility with previous versions of SQL Compare

Snapshots created using SQL Compare versions 3, 4, 5, 6, or 7 can be used in this version of SQL Compare.

However, if the Add WITH ENCRYPTION option was selected when you created a snapshot using SQL Compare version 3, the comparison or deployment may fail when you use the snapshot in this version of SQL Compare.

## Working with scripts folders

A scripts folder is a set of object creation scripts representing a database's schema and (optionally) data.

An object creation script file is created for each object, and different object types are stored within subfolders you can specify.

You can use scripts folders:

- for version control of databases  
For example, you may want to store the script files in a source control system, so that you can track the modified or new objects.
- to compare databases on unconnected SQL Servers

You can create and compare scripts folders only if you are using SQL Compare Professional Edition.

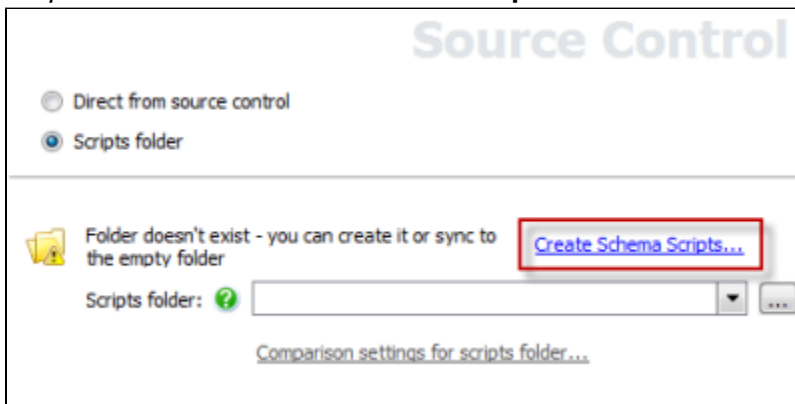
For more information about using scripts folders in source control, see: [Continuous integration for databases using Red Gate SQL tools \(pdf\)](#).

## Creating a scripts folder

To create a new scripts folder:

1. On the **File** menu, select **Create Scripts Folder**.

Alternatively, on the **Project Configuration** dialog box, select *Source Control* as a data source, then select *Scripts folder*. Click **Create Schema Scripts**:



The **Create New Scripts Folder** dialog box is displayed:



2. Under **Data source details**, specify the details of the source for the scripts folder. You can create a scripts folder from a database, backup, snapshot, or another scripts folder.
3. Under **Scripts folder properties**, specify a name and location for the scripts folder.
4. SQL Compare automatically detects the case sensitivity of the data source. If you want to override this, clear the **Auto detect case sensitivity** check box, and select the **Treat items as case sensitive** check box if required.
5. By default, SQL Compare decrypts text objects in SQL Server 2008 and SQL Server 2005 databases that were created using the WITH ENCRYPTION option. When comparing large databases, selecting **Decrypt encrypted objects** can result in slower performance.
6. To customize the folder structure and character encoding of the scripts folder, click **Scripts creation options**. The **Edit Scripts Creation Options** dialog box is displayed, allowing you to specify the directories in which your database objects and static data are saved.

When the data source is a snapshot, the case sensitivity and decryption options are not available.

When the data source is a scripts folder, the decryption option is not available.

7. To finish, click **Create Scripts Folder**.

## Comparing and deploying scripts folders

You can:

- compare a scripts folder with another data source  
See: [Setting data sources](#).
- deploy a scripts folder

When a scripts folder is the target, you can either:

- Create a deployment script to update the database from which the scripts folder was created, or
- Update files in the scripts folder directly

When an object is dropped during deployment, its script file is not deleted.

If a scripts folder is the target, and any of the script files that will be modified are designated as read-only, a warning is displayed. If you click **Yes** to continue, these files will be made writable so that they can be modified. This may occur, for example, when you are working with a source control system that sets files to read-only status in some situations.

For more information, see: [Using the Deploying wizard](#).

## More information about scripts folders

This section provides further information about using object creation scripts as a data source in SQL Compare.

- **White space**  
SQL Server does not always process white space and comments correctly at the beginning and end of the object definition for some objects such as views, stored procedures, and rules. You are therefore recommended to select the **Ignore White Space** option when you use a scripts folder as a data source.
- **Numbered stored procedures**  
Numbered stored procedures are not supported for scripts folders.
- **CLR assemblies**  
SQL Compare can compare object creation scripts that contain CLR assemblies, or include paths to assembly files. When deploying to a scripts folder, SQL Compare will use the hexadecimal content of the assembly.
- **Certificates, symmetric keys, and asymmetric keys**  
These are not supported when you use a scripts folder as a data source.
- **Comments**  
When you select a scripts folder as the target data source, SQL Compare preserves comments in object types such as views and stored procedures. However, this is not possible for non-textual object types such as tables. Comments that are part of a table definition will be lost when the table is modified and the object creation script updated.

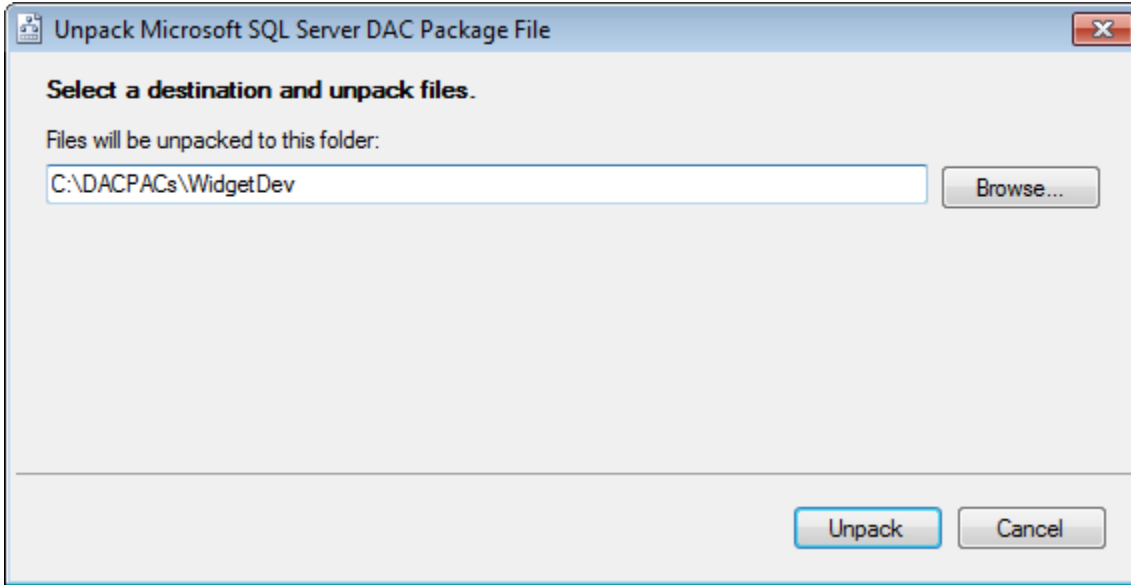
## Using a DACPAC as a data source

DACPAC support is in beta.

You can use SQL Compare to compare data-tier application packages (DACPACs).

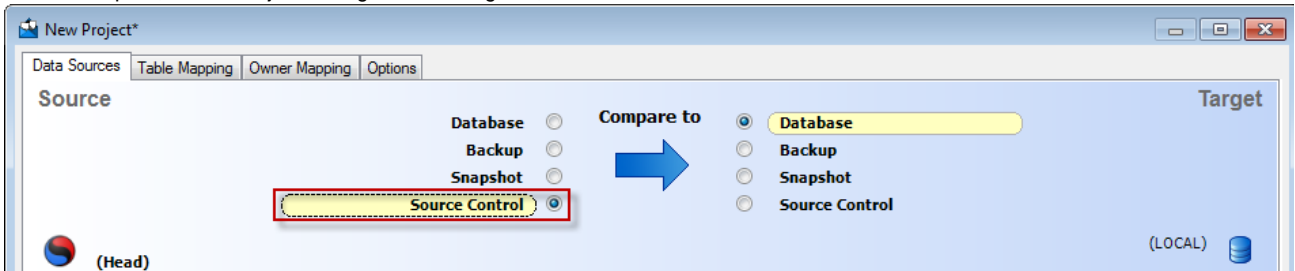
To specify a DACPAC as a [data source](#):

1. In Windows Explorer, right-click the DACPAC file you want to use, and click **Unpack**.  
The **Unpack Microsoft SQL Server DAC Package File** dialog opens.
2. Specify the folder you want the files in the DACPAC to be unpacked to, and click **Unpack**:

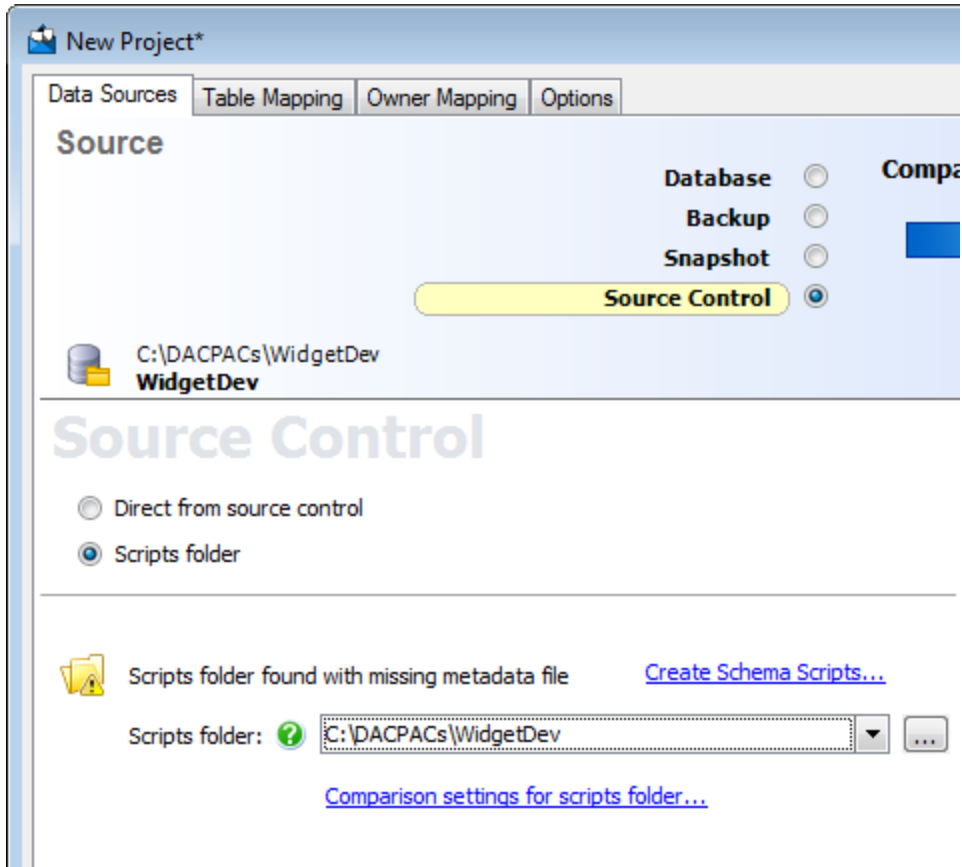


DACPACs contain a *model.sql* file. This is a creation script for all the database objects in the package. SQL Compare uses it when you compare a DACPAC.

3. In SQL Compare, on the Project Configuration dialog box, select **Source Control** as a data source:



4. Select **Scripts folder**, and browse to the folder you unpacked the DACPAC to:



When you specify a DACPAC folder, SQL Compare displays a warning: 'Scripts folder found with missing metadata file'. This is a known issue with the beta DACPAC support, and can be ignored.

If you have any problems using DACPACs with SQL Compare, contact Red Gate support.

# SQL Server Management Studio add-in

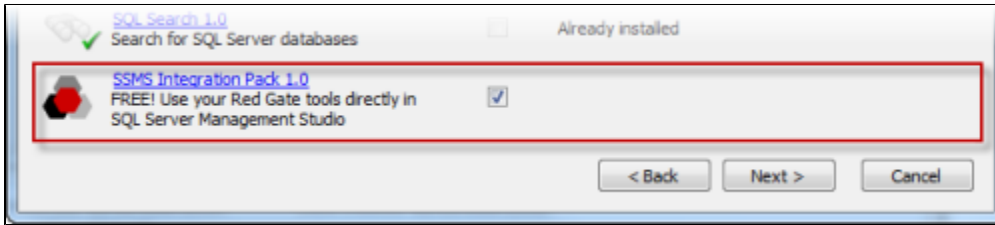
SQL Compare includes an add-in for SQL Server Management Studio (SSMS).

- [Getting started with the add-in](#)
- [Using SQL Compare projects](#)

## Getting started with the add-in

SQL Compare includes a free add-in for SQL Server Management Studio that lets you compare and deploy databases (including versions from source control) from Management Studio.

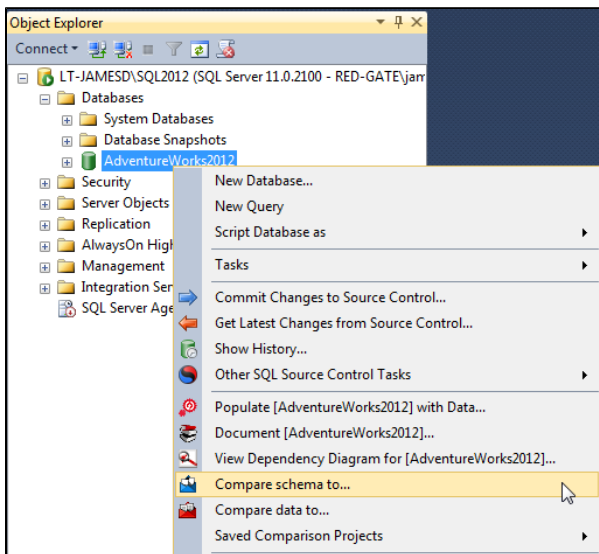
The add-in is bundled with the SQL Compare installer, which you can [download here](#). When you run the installer, make sure the **SSMS Integration Pack** check box is selected:



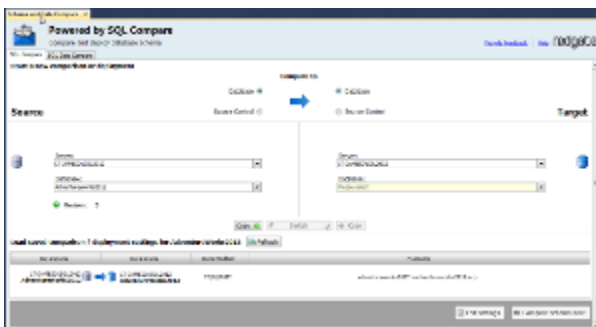
## Comparing databases

To compare two databases from Management Studio:

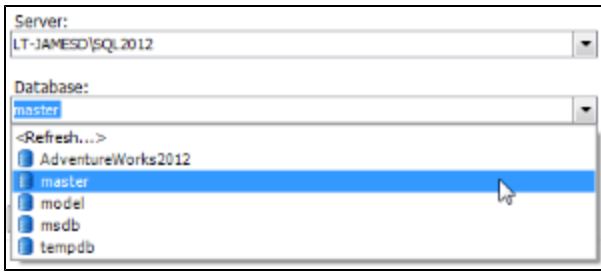
1. In the SQL Server Management Studio Object Explorer, right-click a database you want to compare, and click **Compare schema to**:



The Schema and Data Compare tab is displayed, with the database you selected set as the source in the leftmost pane:



2. In the rightmost pane, select the database you want to set as the target:



To switch the source and target and databases, click



For more information on selecting databases to compare in SQL Compare, see: [Setting data sources](#).

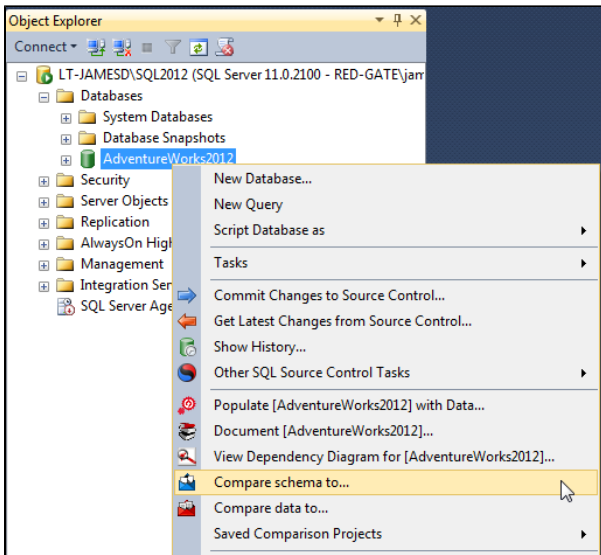
3. Click **Compare schema now**.  
SQL Compare launches, displaying the comparison results.

## Comparing a version from source control

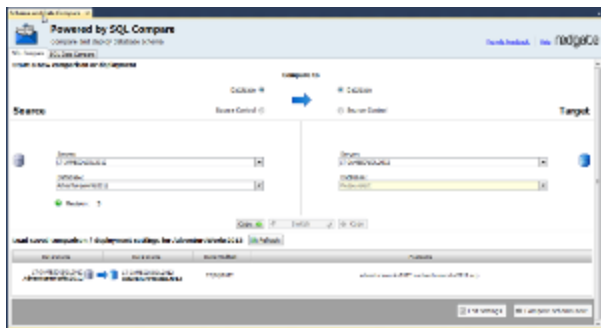
The SQL Server Management Studio add-in enables you to compare and deploy versions of a database linked to [SQL Source Control 1.1](#) or later.

To set a version from source control as a data source:

1. In SQL Server Management Studio Object Explorer, right-click a database you want to compare, and click **Compare schema to**:



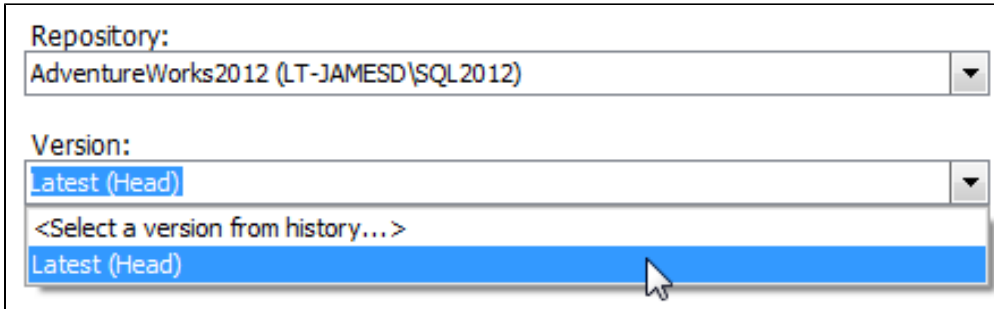
The Schema and Data Compare tab is displayed, with the database you selected set as the source in the leftmost pane:



2. In the upper pane, under Compare To, select **Source Control**:



3. In the **Repository** box, select a database linked to SQL Source Control, or click **<Browse source control...>** to specify a repository URL.
4. In the **Version** box, select a specific revision from the source control history, or select the latest revision:



5. Click **Compare Now**.  
SQL Compare launches, displaying the comparison results.

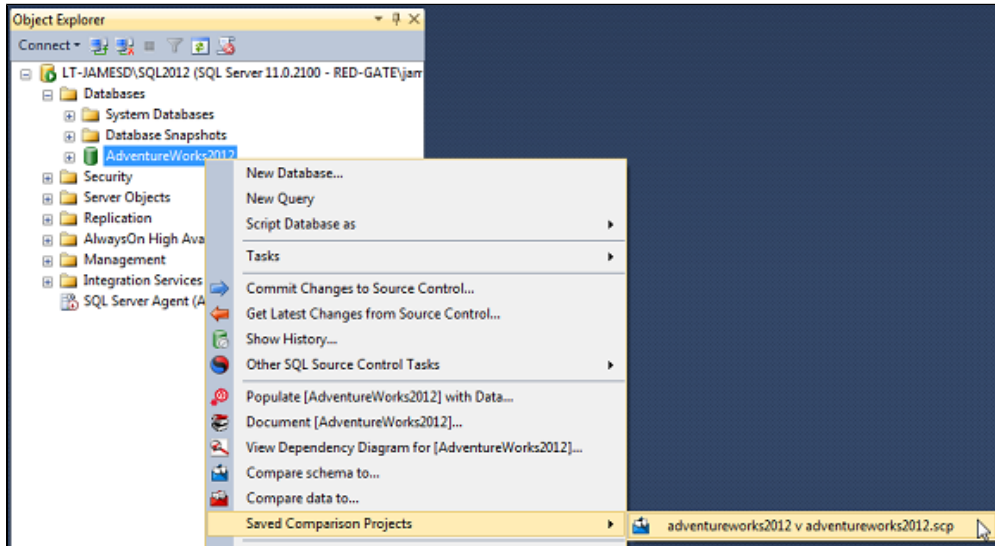


## Using SQL Compare projects

With the SQL Server Management Studio add-in, you can launch SQL Compare projects from SQL Server Management Studio.

### Launching a project from SQL Server Management Studio

In SQL Server Management Studio Object Explorer, right-click a database you want to compare in the project you want to launch. Under **Saved Comparison Projects**, click the project file you want to launch:



SQL Compare launches and opens the project.

# Using the command line

The command line interface provides access to the functionality of SQL Compare. For example, using the command line interface you can:

- automate the comparison and deployment of database schema
- perform scheduled comparisons and deployments
- deploy multiple databases

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 VB, C++ and C#.

## Requirements

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

- A SQL Compare Professional Edition, SQL Developer Bundle, or SQL Toolbelt license  
If you don't have a license, you can use the command line on a trial license 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 command line interface.
- MDAC 2.8 or later

## Command line basics

This page provides information on the following aspects of the SQL Compare command line:

- [Getting help](#)
- [Entering a command](#)
- [Aliases](#)
- [The /Options switch](#)
- [Verbose and quiet switches](#)
- [Redirecting command line output](#)

For details and examples of all of the switches that are available for the SQL Compare command line, see [Switches used in the command line](#)

### Getting help

To display a brief description of SQL Compare, and basic help on all the command line switches, enter:

```
sqlcompare /help
```

For more detailed help enter:

```
sqlcompare /help /verbose
```

This displays a detailed description of each switch and the values it can accept (where applicable), and all exit codes. To output the help in HTML format, enter:

```
sqlcompare /help /verbose /html
```

### Entering a command

When you enter a command, the order of switches is unimportant.

We recommend you follow the Microsoft convention of separating a switch from its values using a colon as shown below.

```
/Out:Output.txt
```

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, don't 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 `/include` and `/exclude` switches are used to include and exclude objects from the comparison and deployment:

```
/Include:table:Product
```

This 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.

Switches and aliases are not case sensitive.

## The /Options switch

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

```
/Options:CaseSensitiveObjectDefinition
```

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

For more information about options, and a list of default option settings, see: [Options used in the command line](#).

## 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 line output

The command line output can be redirect using either the `/Out` switch, or the output redirection features provided by the shell in which you are executing the command.

## Using /Out

You can use the `/out` switch to specify the file to which you want output directed:

```
sqlcompare ... /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.

## Other redirection

From the standard command prompt provided by Windows, you can redirect output to a file as follows:

```
sqlcompare ... > 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, enter the following:

```
sqlcompare... >> existinglog.txt
```

This adds the output to the existing file content, without data being lost.

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 more on this topic see: [Using XML to specify command line arguments](#).

## Integrating the command line with applications

To integrate the SQL Compare command line into automated process within your environment, you must purchase a [DLM Automation Suite license](#) (previously called the SQL Automation Pack) for each instance of the automated process.

The files you need to bundle into your application installer are listed below. The files should be installed in the same folder as your application.

For all SQL Compare functions:

- SQLCompare.exe
- SQLCompare.exe.config
- System.Threading.dll

For interfacing with SQL Source Control:

- RedGate.SOCCompareInterface.dll

For creating reports:

- SQLCompareExcel.xslt
- SQLCompareInteractiveReportTemplate.xslt
- SQLCompareSimpleHTMLReport.xslt
- the Reporting folder

For reading backups only:

- System.Data.SQLite.dll  
This file is located in the *SQLite* subfolder of the SQL Compare installation location.

For reading encrypted backups:

- RedGate.BackupReader.CryptoHelper.dll

For reading compressed backups:

- zlib1.dll

## Simple examples using the command line

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

For more detailed information on using the command line, see:

- [Example - selecting single tables for comparison](#)
- [Example - selecting tables with unrelated names](#)
- [Switches used in the command line](#)

## Comparing and deploying database schema

To compare the database structure of *WidgetStaging* with *WidgetProduction*:

```
sqlcompare /Database1:WidgetStaging /Database2:WidgetProduction
```

To compare the database structure of *WidgetStaging* with *WidgetProduction*, and deploy the databases by updating *WidgetProduction*:

```
sqlcompare /Database1:WidgetStaging /Database2:WidgetProduction /Synchronize
```

## Using an existing SQL Compare project from the command line

This is useful, for example, as you can't choose or create a [custom filter](#) from the command line, and specifying complex object selections using a regular expression can be unwieldy.

To use a project you have saved as "*widgets.scp*" from the command line:

```
sqlcompare /project:"C:\SQLCompare\Projects\Widgets.scp"
```

- When you use a project, all objects that were selected for comparison when you saved the project are automatically included.
- When you use the command line, your project option selections are ignored and the defaults are used. Use */options* to specify any additional options you want to use with a command line project.  
For more information, see: [Options used in the command line](#).
- If you want to include or exclude objects from an existing project, you must modify your selection using the graphical user interface.  
You can't use the */include* and */exclude* switches with */project*.

## Scheduling or automating a comparison or deployment

You can use the Microsoft Windows Scheduled Task wizard to schedule a comparison by creating a script to run the comparison.

The following example compares the structure of *WidgetStaging* and *WidgetProduction*, and outputs the results as the file *log\_file.txt*

First create the script:

```
cd "C:\Program Files\Red Gate\SQL Compare 8" sqlcompare  
/db1:WidgetStaging /db2:WidgetProduction >> C:\log_file.txt
```

Next save the script as a *.bat* file. You specify the *.bat* file as the program to run from within the Scheduled Task wizard by browsing to it.

To schedule a deployment of the two databases, updating the database *WidgetProduction*, you would create the script:

```
cd "C:\Program Files\Red Gate\SQL Compare 8" SQLCompare /db1:WidgetStaging
/db2:WidgetProduction /Synchronize
```

In these examples MS-DOS batch scripting is used, a basic scripting language that is supported on all versions of Windows. If preferred, you could use VBScript, JScript, PHP, Perl, Python or any other scripting language of your choice.

## Creating a scripts folder

You can create a folder of object creation scripts representing the schema of a data source. To export *WidgetProduction* to a scripts folder:

```
sqlcompare /Database1:WidgetProduction /Makescripts:"C:\WidgetProductionScripts"
```

If the folder does not already exist, it is created. All the subfolders containing different object types in the schema are automatically created beneath the specified folder. If the folder does exist, it must be empty or the export will fail.

For more information on scripts folders, see [Working with scripts folders](#).

## Creating a snapshot

You can create a SQL Compare snapshot representing the schema of a data source. To export *WidgetProduction* to a snapshot:

```
sqlcompare /Database1:WidgetProduction /Makesnapshot:"C:\Snapshots\WP_1.snp"
```

For more information on SQL Compare schema snapshots, see: [Working with snapshots](#).

## Using a scripts folder or snapshot as a data source

To compare the *WidgetProduction* scripts folder with the *WidgetStaging* database:

```
sqlcompare /Scripts1:"C:\WidgetProductionScripts" /Database2:WidgetStaging
```

To compare the *WidgetStaging* database with the *WidgetProduction* scripts folder and deploy the scripts

```
sqlcompare /Database1:WidgetStaging /Scripts2:"C:\WidgetProductionScripts"
/Synchronize /Force
```

The */force* switch specifies that any read-only files in the scripts folder that need to be edited during deployment will be made writable. If you do not include the */force* switch and read-only files need to be modified, the deployment will fail and an error message will be displayed.

To compare two snapshots of *WidgetStaging*:

```
sqlcompare /Snapshot1:"C:\Snapshots\WidgetProd_1.snp"
/Snapshot2:"C:\Snapshots\WidgetProd_2.snp"
```

To output the deployment SQL script, for example for auditing purposes, and overwrite the file if it already exists:

```
sqlcompare /database1:WidgetStaging /database2:WidgetProduction
/scriptfile:"C:\SQLScripts\Widgets.sql" /force
```



## Using a backup as a data source

To compare a backup of *WidgetDev* with *WidgetLive*:

```
sqlcompare /Backup1:"D:\MSSQL\BACKUP\WidgetDev_20080807_143235.sqb" /db2:WidgetLive
```

If you are comparing a backup set that contains multiple files, use the `/backupset1` switch to specify the files which make up the first backup set, and use the `/backupset2` switches to specify the files which make up the second:

```
sqlcompare /Backup1:"D:\MSSQL\BACKUP\WidgetDev.bak" /Backupset1:"2008-09-23 Full Backup" /db2:WidgetLive
```

If the backup set switches are not specified, SQL Compare uses the latest backup set.

To specify more than one backup file, separate the file names using semicolons.

```
sqlcompare /Backup1:"D:\WidgetDev_Full.bak";"D:\WidgetDev_Diff.bak" /db2:WidgetDevelopment
```

For encrypted backups that have been created using SQL Backup, use the `/BackupPasswords1` and `/BackupPasswords2` switches to specify the passwords; when there is more than one backup password, separate the passwords using semicolons.

```
sqlcompare /Backup1:"D:\MSSQL\BACKUP\WidgetDev.sqb" /BackupPasswords1:Pa$$w0rd /db2:WidgetLive
```

If you are comparing a differential backup, you must also specify the associated full backup.

For more information, see: [Working with backups](#).

## 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 */Include* and */Exclude* switches, use the following format:

```
<include>objecttype:RegularExpression</include>
```

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

```
sqlcompare /Argfile:XMLfilename.xml
```

When using an XML file

- you can't specify any other switches on the command line except */verbose* or */quiet*
- multiple options should be separated with commas:

```
<options>n,oc,t</options>
```

## Examples

Below are some examples of XML files that can be used with SQL Compare. The command line versions of the examples (using aliases) are also provided for comparison. To migrate changes in the XML examples, use the *<synchronize/>* element.

In all these examples, you can use a scripts folder or snapshot in place of a database. Use *<scripts1>* and *<scripts 2>* in the XML file for a scripts folder, and *<snapshot1>* and *<snapshot2>* for a snapshot. In the command line, replace */db* with */scr* for a scripts folder or */snp* for a snapshot.

### **To compare all objects in two local databases (Windows authentication):**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<database2>SecondDatabaseName</database2>
</commandline>
```

Using the command line:

```
SQLCompare /db1:FirstDatabaseName /db2:SecondDatabaseName
```

**To compare all objects in databases on different hosts:**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<server1>Hostname1</server1>
<database2>SecondDatabaseName</database2>
<server2>Hostname2</server2>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /s1:Hostname1 /db2:SecondDatabaseName /s2:Hostname2
```

**To compare all objects in two databases using SQL Server authentication:**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<username1>Username1</username1>
<password1>Password1</password1>
<database2>SecondDatabaseName</database2>
<username2>Username2</username2>
<password2>Password2</password2>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /u1:Username1 /p1:Password1 /db2:SecondDatabaseName
/u2:Username2 /p2:Password2
```

**To retrieve verbose output of the differences between two databases:**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<database2>SecondDatabaseName</database2>
<verbose/>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /db2:SecondDatabaseName /Verbose
```

**To migrate schema changes from the first database to the second database:**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<database2>SecondDatabaseName</database2>
<synchronize/>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /db2:SecondDatabaseName /sync
```

**To compare only tables containing the word "Product":**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<database2>SecondDatabaseName</database2>
<include>Table</include>
<include>Table:Product</include>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /db2:SecondDatabaseName /Include:table
/Include:table:\[Product\]
```

**To compare only tables containing the word "Product" except for the "ProductHistory" table:**

Using an XML file:

```
<?xml version="1.0"?>
<commandline>
<database1>FirstDatabaseName</database1>
<database2>SecondDatabaseName</database2>
<include>Table</include>
<include>Table:Product</include>
<exclude>Table:ProductHistory</exclude>
</commandline>
```

Using the command line:

```
sqlcompare /db1:FirstDatabaseName /db2:SecondDatabaseName /Include:table
/Include:table:\[Product\] /Exclude:table:\[ProductHistory\]
```

## Scripts folder database information file

When you create a scripts folder, an XML file containing some basic details about the structure of the database is created. The file is called *RedGateDatabaseInfo.xml* and is stored in the main scripts folder at the top level.

The file contains the following information, with the defaults as shown:

```
<?xml version="1.0" encoding="utf-16">
<DatabaseInformation>
<DefaultCollation>Latin1_General_CI_AS</DefaultCollation>
<DefaultSchema>dbo</DefaultSchema>
<DefaultUser>dbo</DefaultUser>
<DefaultFilegroup>PRIMARY</DefaultFilegroup>
<DatabaseVersion>9</DatabaseVersion>
</DatabaseInformation>
```

If you have your own set of SQL creation scripts that you want to use in SQL Compare, you can specify the settings for your schema by editing this file. If the file does not exist when you run the comparison, SQL Compare creates it, using the default values.

## Example - selecting single tables for comparison

This shows how you can use the command line or an XML arguments file to select a single table for comparison.

In this example, the databases contain the following tables (amongst others):

- Product
- ProductCategory
- ProductCostHistory
- ProductDealerPriceHistory
- SpecialOfferProduct

You are interested only in the schema differences between the *Product* tables in two different versions of your database; you are not interested in any of the other tables, or any other objects in the database.

You can replace the database (*/db*) in this example with either a scripts folder (*/scr*) or a snapshot (*/sn*) and the example will still work correctly. For more information, see: [Simple examples using the command line](#).

## Using the command line

To specify the table to include, use the */include* switch:

```
sqlcompare /db1:Products1 /db2:Products2 /Include:table /Include:table:\[Product\  
/verbose
```

where:

*/db1:Products1* specifies that you want to compare the database *Products1*

*/db2:Products2* specifies that you want to compare the database *Products2*

*/Include:table* specifies that you want to compare only tables; you do not want to compare other objects such as views, stored procedures, and so on. If you omit this argument, SQL Compare compares all tables that have names that match the second */Include* switch and all other objects in the databases.

To specify more than one object type for inclusion, use multiple */include* switches. For example, to include only tables and views, enter:

```
/Include:table /Include:view
```

*/Include:table:\[Product\  
] specifies that you want to compare only the table that has a name that includes the string *Product**

Because you use .NET standard regular expressions to define the */Include* and */Exclude* arguments, you must escape the square brackets ( `[]` ) with the backslash character ( `\` ). Regular expression syntax is beyond the scope of this online help; refer to your Microsoft .NET framework documentation for more information.

You must include the brackets ( `[]` ) in the string; if you specify the argument without the brackets, */Include:table:Product*, the *ProductCategory* table is included because it contains the string *Product*. The full SQL Server table names are qualified by the owner name in SQL Server 2000, and the schema name in SQL Server 2005/2008, and include brackets. For example (in SQL Server 2000):

```
[dbo].[Product]
```

```
[dbo].[ProductCategory]
```

and so on. Therefore, the brackets indicate that you are specifying the full table name. To include the owner (or schema) name in the regular expression, you would need also to escape the dot ( `.` ):

```
/Include:table:\[dbo\  
]\.[Product]
```

The pipe character ( `|` ) in a regular expression is interpreted as a logical OR. The character must be escaped by the caret character ( `^` ), to prevent the operating system shell from interpreting it as the pipe operator. (If you want to use the caret character itself as part of your regular expression, it must be escaped by a second caret.)

*/Verbose* specifies that you want to display detailed information about differences between objects

## Using XML

You can use XML like this:

```
<?xml version="1.0"?>
<commandline>
  <database1>Products1</database1>
  <database2>Products2</database2>
  <verbose/>
  <include>Table</include>
  <include>Table:\[Product\]</include>
</commandline>
```

To execute the comparison using the XML file, enter the following command:

```
sqlcompare /Argfile:XMLFileName.xml
```

where *XMLFileName* is the name of the XML file.

## Example - selecting tables with unrelated names

This example illustrates how to select a number of individual tables for comparison when their names are not related in any way.

In this example, the databases contain the following tables:

- Product
- Supplier
- ProductCategory
- SpecialOffer
- Customer
- Order
- Invoice

You are interested only in the schema differences between the *Product*, *Customer*, *Order*, and *Invoice* tables in two different versions of your database, *Customers1* and *Customers2*; you are not interested in any of the other tables, or any other objects in the databases.

### Using the command line

To specify the list of tables to include, you use the */include* switch. You could use an *include* switch for each table that you want to compare. However, this could get unwieldy if you have a long list of tables. Instead, you can use the pipe character ( | ) to separate the table names:

```
sqlcompare /db1:Customers1 /db2:Customers2
/Include:table /Include:table:\[Product\]^|Customer^|Order^|Invoice
```

where:

*/db1:Customers1* specifies that you want to compare the database *Customers1*

*/db2:Customers2* specifies that you want to compare the database *Customers2*

*/Include:table* specifies that you want to compare only tables; you do not want to compare other objects such as views, stored procedures, and so on. If you omit this argument, SQL Compare compares all tables that match the second */Includeswitch* and all other objects in the databases.

To specify more than one object type for inclusion, use multiple */Include* switches. For example, to include only tables and views, enter:

```
/Include:table /Include:view /Include:table:\[Product\]^|Customer^|Order^|Invoice
```

specifies that you want to compare only the tables that have a name that includes the strings *[Product]*, or *Customer*, or *Order*, or *Invoice*

Because you use .NET standard regular expressions to define the */Include* and */Exclude* arguments you must escape the square brackets ( [ ] ) with the backslash character ( \ ). Regular expression syntax is beyond the scope of this online help; refer to your Microsoft .NET framework documentation for more information.

You must include the brackets ( [ ] ) in the string; if you specify the argument without the brackets, */Include:table:Product*, the *ProductCategory* table is included because it contains the string *Product*. The full SQL Server table names are qualified by the owner name in SQL Server 2000, and the schema name in SQL Server 2005/2008, and include brackets. For example (in SQL Server 2000):

```
[dbo].[Product]
[dbo].[ProductCategory]
```

and so on. Therefore, the brackets indicate that you are specifying the full table name. To include the owner (or schema) name in the regular expression, you also need to escape the dot ( . ):

```
/Include:table:\[dbo\]\. \[Product\]
```

The pipe character ( | ) in a regular expression is interpreted as a logical OR. The character must be escaped by the caret character ( ^ ), to prevent the operating system shell from interpreting it as the pipe operator.



If you want to use the caret character itself as part of your regular expression, it must be escaped by a second caret.

## Using XML

You can use XML like this:

```
<?xml version="1.0"?>
<commandline>
  <database1>Customers1</database1>
  <database2>Customers2</database2>
  <sync/>
  <include>Table</include>
  <include>Table:\[Product\]|Customer|Order|Invoice</include>
</commandline>
```

The pipe character (|) (and other operating system operators) do not have to be escaped by the caret character (^) when they are specified in the XML file.

To execute the comparison using the XML file, enter the following command, where *XMLFileName* is the name of the XML file:

```
sqlcompare /Argfile:XMLFileName.xml
```

## Switches used in the command line

This is a list of switches you can use with the SQL Compare command line.

- The first data source ( /db1, /b1, etc) is the *source*.
- The second data source ( /db2, /b2, etc) is the *target*
- The command line syntax of previous versions of SQL Compare is considered deprecated, but is still supported. For example, in SQL Compare 7, the alias for /BackupSet1 was /bs1. In SQL Compare 8 or later, the alias is now /bks1. You can continue to use /bs1 in SQL Compare 8 or later, but a message is displayed informing you of the new alias. Deprecated command line syntax will be removed in a future release.

### /AbortOnWarnings

Alias: /aow

Specifies that SQL Compare won't run a deployment if there are any serious deployment warnings. If you don't specify this switch, SQL Compare will ignore warnings and run the deployment.

### Arguments

None	Don't abort on warnings
Medium	Abort on medium or high warnings
High	Abort on high warnings

The default is None. If you use this switch and there are deployment warnings, exit code 61 is displayed.

For more information on warnings in SQL Compare, see [Deployment Warnings](#).

### /activateSerial:<serial number>

- This switch is case sensitive.
- An internet connection is required to activate SQL Compare from the command line. For information about how to activate manually without an internet connection, see [Activating](#).

Attempts to activate SQL Compare.

You can specify a SQL Compare Professional serial number, or a serial number for bundle such as the SQL Developer Bundle.

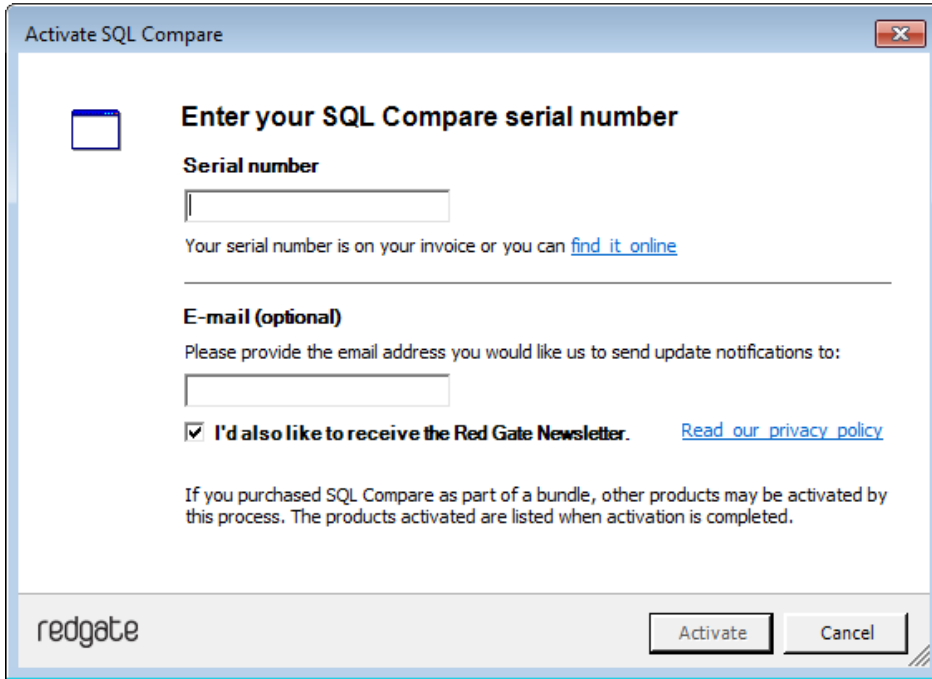
#### Example

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

If you run the switch without specifying a serial number, the **Activate SQL Compare** dialog box is displayed:

### Contents

- /AbortOnWarnings
- /activateSerial:<serial number>
- /Argfile:<file path>
- /Assertidental
- /Backup1:<file path1>;<file path2>;...;<file pathN>
- /Backup2:<file path1>;<file path2>;...;<file pathN>
- /BackupPasswords1:<password1>,<password2>,...,<passwordN>
- /BackupPasswords2:<password1>,<password2>,...,<passwordN>
- /BackupSet1:<backup set>
- /BackupSet2:<backup set>
- /Database1:<database name>
- /Database2:<database name>
- /deactivateSerial
- /Exclude:<object type>:<regular expression>
- /Filter
- /Force
- /Help
- /HTML
- /IgnoreParserErrors
- /IgnoreSourceCaseSensitivity
- /Include:<object type>:<regular expression>
- /LogLevel:<level>
- /MakeScripts:<folder>
- /MakeSnapshot:<file name>
- /MigrationsFolder:<folder>
- /Options:<option1>,<option2>,<option3>
- /Out:<file path>
- /OutputProject:<file path>



If you're using the SQL Compare command line on a server, you need a DLM Automation Suite license (previously called SQL Automation Pack).

For information about how to activate the DLM Automation Suite, see [Activating](#).

### **/Argfile:<file path>**

Runs a file containing an XML argument specification.

```

Example
sqlcompare /Argfile:XMLFileName.xml

```

For more information, see [Using XML to specify command line arguments](#).

### **/Assertidentical**

When */assertidentical* is specified, SQL Compare will return an exit code of 0 if the objects being compared are identical. If they aren't identical, it will return exit code 79.

### **/Backup1:<file path1>;<file path2>;...;<file pathN>**

Alias: */b1*

Specifies the backup to be used as the source. You must add all of the files making up the backup set you want to compare.

```

Example
sqlcompare /Backup1:D:\BACKUPS\WidgetStaging.bak
/db2:WidgetStaging

```

To specify more than one backup file, the file names are separated using semicolons.

- /OutputWidth:<columns>
- /Password1:<password>
- /Password2:<password>
- /Project:<file path>
- /Quiet
- /Report:<file path>
- /reportAllObjectsWithDifferences
- /ReportType:<report type>
- /Revision1:<revision>
- /Revision2:<revision>
- /ScriptFile:<file path>
- /Scripts1:<folder>
- /Scripts2:<folder>
- /ScriptsFolderXML:<file path>
- /Server1:<server name>
- /Server2:<server name>
- /ShowWarnings
- /Snapshot1:<filename>
- /Snapshot2:<filename>
- /Sourcecontrol1
- /Sourcecontrol2
- /Synchronize
- /SyncScriptEncoding:<script encoding>
- /Tempinstance:<tempinstance>
- /TransactionIsolationLevel:<transaction isolation level>
- /UserName1:<username>
- /UserName2:<username>
- /Verbose
- /VersionUserName1:<username>
- /VersionUserName2:<username>
- /VersionPassword1:<password>
- /VersionPassword2:<password>

### Example

```
sqlcompare /Backup1:D:\BACKUPS\WidgetDev_Full.bak;  
D:\BACKUPS\WidgetDev_Diff.bak /db2:WidgetDev
```

For more information, see [Working with backups](#).

- **Deprecated options**
  - /AllowIdentical Databases
  - /IncludeIdentical:<IncludeIdentical>
  - /MigrationsFolderXML:<file path>

### **/Backup2:<file path1>;<file path2>;...;<file pathN>**

Alias: */b2*

Specifies the backup to be used as the target. You must add all of the files making up the backup set you want to compare.

### Example

```
sqlcompare /db1:WidgetStaging  
/Backup2:D:\BACKUPS\WidgetStaging.bak
```

### **/BackupPasswords1:<password1>,<password2>,....,<passwordN>**

Alias: */b1*

Specifies the password for the source backup.

### Example

```
sqlcompare /Backup1:D:\BACKUPS\WidgetStaging.bak  
/BackupPasswords1:P@ssw0rd /db2:WidgetProduction
```

### **/BackupPasswords2:<password1>,<password2>,....,<password1N>**

Alias: */b2*

Specifies the password for the target backup:

### Example

```
sqlcompare /db1:WidgetStaging  
/Backup2:D:\BACKUPS\WidgetProduction.bak  
/BackupPassword2:P@ssw0rd
```

### **/BackupSet1:<backup set>**

Alias: */bks1*

If you are comparing a backup set that contains multiple files, use the /BackupSet1 switch to specify the files which make up the source backup set, and use the /BackupSet2 switches to specify the files which make up the target.

### Example

```
sqlcompare /Backup1:"D:\MSSQL\BACKUP\WidgetDev.bak" /BackupSet1:"2008-09-23 Full Backup" /db2:WidgetLive
```

If the backup set switches aren't specified, SQL Compare uses the latest backup set.

To specify more than one backup file, the file names are separated using semi-colons.

### Example

```
sqlcompare /Backup1:D:\BACKUPS\WidgetDev_Full.bak; "D:\BACKUPS\WidgetDev_Diff.bak" /db2:WidgetDevelopment
```

For encrypted backups that have been created using SQL Backup, use the /BackupPasswords1 and /BackupPasswords2 switches to specify the passwords; when there is more than one password, the passwords are separated using semi-colons.

### Example

```
sqlcompare /Backup1:D:\BACKUPS\WidgetDev.sqb /BackupPassword1:Pa$$w0rd /db2:WidgetLive
```

## /BackupSet2:<backup set>

Alias: */bks2*

Specifies which backup set to use for the target backup.

### Example

```
sqlcompare /db1:WidgetProduction /BackupSet2:"2008-09-23 Full Backup"
```

## /Database1:<database name>

Alias: */db1*

Specifies a database to use as the source.

### Example

```
sqlcompare /Database1:WidgetStaging /Database2:WidgetProduction
```

## /Database2:<database name>

Alias: */db2*

Specifies a database to use as the target.

## /deactivateSerial

This switch is case sensitive.

Attempts to deactivate the application. An internet connection is required to deactivate the product.

## /Exclude:<object type>:<regular expression>

We recommend using filters instead of */Exclude*. For more information, see [Using filters](#).

### Arguments:

• <i>Additional</i>	only those objects that aren't present in the source (eg <i>/db1</i> )
• <i>Missing</i>	only those objects that aren't present in the target (eg <i>/db2</i> )
• <i>Different</i>	only those objects that aren't present in both data sources, but are different.
• <i>Identical</i>	identical objects in the command line output and any generated reports.
• <i>User specified</i>	objects you specify with a regular expression (eg <i>/include:Table:WidgetPurchases</i> )

To specify the list of objects to exclude, use the */exclude* switch.

### Example

```
sqlcompare /db1:Customers1 /db2:Customers2 /exclude:table
```

*/exclude:table* specifies that you don't want to compare tables; you only want to compare other objects such as views, stored procedures, and so on.

To specify more than one object or object type for exclusion use multiple */exclude* switches. For example, to exclude only tables and views.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /exclude:table:WidgetReferences  
/exclude:view
```

You can't use */exclude* with the */include* and */project* switches.

For a more detailed example using the */include* and */exclude* switches, see [Selecting tables with unrelated names](#).

## /Filter

Alias: */fr*

Specifies a custom filter to select objects for deployment.

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /sync
```

```
  /filter:MarketingViewsOnly.scpf
```

You can set up a filter to include or exclude objects based on their type, name, and owner (schema) name.

This is useful, for example, if you want to create complex selection rules without using regular expressions.

- filters are set up in the graphical user interface
- filters are saved with the extension *.scpf*
- */filter* can't be used with */Include* or */Exclude*
- if you use */filter* with */project*, the filter you specify overrides any filter used in the project

For more information, see [Using filters](#).

## ***/Force***

Alias: */f*

This forces the overwriting of any output files that already exist. If this switch isn't used and a file of the same name already exists, the program will exit with the exit code indicating an IO error.

## ***/Help***

Alias: */?*

Displays the list of switches in the command line with basic descriptions.

If */help* is used with any switches except */verbose*, */html*, */out*, */force* or */outputwidth* then those extra switches will be ignored; the help message will be printed and the process will end with exit code 0.

## ***/HTML***

Outputs the help text as HTML. Must be used with the */help* switch.

## ***/IgnoreParserErrors***

If SQL Compare encounters any high level errors when parsing a scripts folder, it will exit with an error code of 62.

Use */ignoreParserErrors* to force SQL Compare to continue without exiting.

## ***/IgnoreSourceCaseSensitivity***

When you are creating a scripts folder using */makescripts*, SQL Compare automatically detects the case sensitivity of the data source.

Use */ignoreSourceCaseSensitivity* to disable automatic detection of case sensitivity.

## ***/Include:<object type>:<regular expression>***

We recommend using filters instead of */include*. For more information, see [Using filters](#).

### **Arguments:**

• <i>Additional</i>	only those objects that aren't present in the source (eg <i>/db1</i> )
• <i>Missing</i>	only those objects that aren't present in the target (eg <i>/db2</i> )
• <i>Different</i>	only those objects that are present in both data sources, but are different.
• <i>Identical</i>	identical objects in the command line output and any generated reports.
• <i>User specified</i>	objects you specify with a regular expression (eg <i>/include:Table:WidgetPurchases</i> )
• <i>StaticData</i>	static data in a source-controlled database or a scripts folder

This switch is used to specify the list of objects to include. You can use an `/include` switch for each object that you want to compare. However, this can be unwieldy if there is a long list. Instead, you can use the pipe character ( `|` ) to separate the table names:

```
sqlcompare /db1:Customers1 /db2:Customers2 /include:table
           /include:table:\[Product\]^|Customer^|Order^|Invoice
```

For more detailed information on using the `/include` switch, see [Selecting tables with unrelated names](#).

## **`/LogLevel:<level>`**

Alias: `/log`

Creates a log file with a specified minimum log level.

Log files collect information about the application while you are using it. These files are useful to us if you have encountered a problem. For more information, see [Logging and log files](#).

### **Arguments:**

• <i>None</i>	Disables logging
• <i>Error</i>	Reports serious and fatal errors
• <i>Warning</i>	Reports warning and error messages
• <i>Verbose</i>	Reports all messages in the log file

The default is `None`.

### **Example**

```
sqlcompare /db1:WidgetStaging /makeScripts: D:\Scripts Folder /logLevel:Verbose
```

You must use `/logLevel` each time you want a log file to be created.

## **`/MakeScripts:<folder>`**

Alias: `/mkscr`

Creates a scripts folder from the data source.

### **Example**

```
sqlcompare /db1:WidgetStaging /makeScripts:"C:\Scripts Folders\Widget staging
scripts"
```

If the folder already exists an error will occur. To merge scripts into an existing scripts folder, compare them with that folder and use the `/synchronize` switch.



### Example

```
sqlcompare /scr1:"C:\Scripts Folders\Widget dev scripts" /scr2:"C:\Scripts  
Folders\Widget staging scripts" /synchronize
```

For more information, see [Working with scripts folders](#).

### **/MakeSnapshot:<file name>**

Alias: */mksnap*

Creates a snapshot from the data source.

### Example

```
sqlcompare /db1:WidgetStaging /makeSnapshot:"C:\Widget  
Snapshots\StagingSnapshot.snp"
```

If the file already exists an error will occur, unless you have also used the */force* switch.

### **/MigrationsFolder:<folder>**

Alias: */mf*

The path to a directory containing migration scripts you want to use.

### **/Options:<option1>,<option2>,<option3>**

Alias: */o*

Applies the project configuration options used during comparison or deployment:

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
/options:Default,IgnoreWhiteSpace
```

For a detailed list of these options see [Options used in the command line](#).

### **/Out:<file path>**

Redirects console output to the specified file.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /out:C:\output file
```

### **/OutputProject:<file path>**

Alias: */outpr*

Writes the settings used for the comparison to the specified SQL Compare project file.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
/options:Default,IgnoreWhiteSpace /outputProject:"C:\WidgetProject.scp"
```

This also generates a SQL Compare project file. These files end with a .scp extension. If the file already exists an error will occur, unless you have also used the */force* switch.

### **/OutputWidth:<columns>**

Forces the width of console output.

This can be used to ensure that database object names etc aren't truncated, and that SQL script lines aren't wrapped or broken. This is particularly useful when redirecting output to a file as it allows you to overcome the limitations of the default console width of 80 characters.

### **/Password1:<password>**

Alias: */p1*

The password for the source database.

You must also provide a username. If you don't specify a username and password combination, integrated security is used.

### Example

```
sqlcompare /db1:WidgetStaging /userName1:User1  
/password1:P@ssw0rd /db2:WidgetProduction /userName2:User2 /password2:Pa$$w0rd
```

This switch is only used if the source is a database. If the source is a backup, use */backupPasswords1*

### **/Password2:<password>**

Alias: */p2*

The password for the target database.

### **/Project:<file path>**

Alias: */pr*

Uses a SQL Compare project (.scp) file for the comparison.

To use a project you have saved as "widgets.scp" from the command line:

### Example

```
sqlcompare /project:"C:\SQLCompare\Projects\Widgets.scp"
```

- When you use a project, all objects that were selected for comparison when you saved the project are automatically

- included.
- When you use the command line, your project option selections are ignored and the defaults are used. Use */options* to specify any additional options you want to use with a command line project. For more information, see [Options used in the command line](#).
- If you want to include or exclude objects from an existing project, you must modify your selection using the graphical user interface.  
You can't use the */include* and */exclude* switches with */project*.

The */project* switch is useful, for example, as you can't specify a [custom filter](#) in the command line, and specifying complex object selections using a regular expression can be unwieldy.

For more information on using projects, and what a project contains, see [Working with projects](#).

## **/Quiet**

Alias: */q*

Quiet mode: no output.

## **/Report:<file path>**

Alias: */r*

Generates a report and writes it to the specified file.

The type of report is defined by the */reportType* switch. If the file already exists an error will occur, unless you have used the */force* switch:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction
/report:"C:\reports\WidgetReport.html" /reportType:Simple
```

## **/reportAllObjectsWithDifferences**

Alias: */rad*

Includes all objects with differences in the reports, rather than all selected objects.

## **/ReportType:<report type>**

Alias: */rt*

### **Arguments:**

• <i>XML</i>	Simple XML report
• <i>Simple</i>	Simple HTML report
• <i>Interactive</i>	Interactive HTML report
• <i>Excel</i>	Microsoft Excel spreadsheet

This switch defines the file format of the report produced by the */Report* switch. The default is XML.

### **Example**

```
sqlcompare /db1:WidgetStaging
/db2:WidgetProduction /report:"C:\reports\WidgetReport.html" /reportType:Simple
```

For more information, see [Exporting the comparison results](#).

## **/Revision1:<revision>**

Alias: */r1*

Specifies the source control revision of the source database. To specify a revision, the database must be linked to SQL Source Control. To specify the latest version, type: *HEAD*

Specifying a revision other than HEAD is only supported with TFS, SVN and Vault. If you're using another source control system, we recommend checking the revision out to a local folder and using the */Scripts1* switch.

The following example compares revision 3 of WidgetStaging with the latest revision of WidgetProduction:

```
sqlcompare /db1:WidgetStaging /revision1:3 /db2:WidgetProduction /revision2:HEAD
```

## **/Revision2:<revision>**

Alias: */r2*

Specifies the source control revision of the target database. To specify a revision, the database must be linked to SQL Source Control.

## **/ScriptFile:<file path>**

Alias: */sf*

Generates a SQL script to migrate the changes which can be executed at a later time. If the file already exists an error will occur, unless you use the */force* switch.

### **Example**

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /scriptFile:"C:\Scripts Folder\WidgetSyncScript.sql"
```

*/Scriptfile* can be used when the target ( */db2*, */scr2*, */sn2* ) is a database, a snapshot, or a scripts folder.

If the target is a snapshot or a scripts folder, the generated script modifies a database with the schema represented by that snapshot or scripts folder.

## **/Scripts1:<folder>**

Alias: */scr1*

Specifies the scripts folder to use as the source.

### **Example**

```
sqlcompare /scripts1:"C:\Scripts Folder\WidgetStagingScript" /db2:WidgetProduction
```

## **/Scripts2:<folder>**

Alias: */scr2*

Specifies the scripts folder to use as the target.

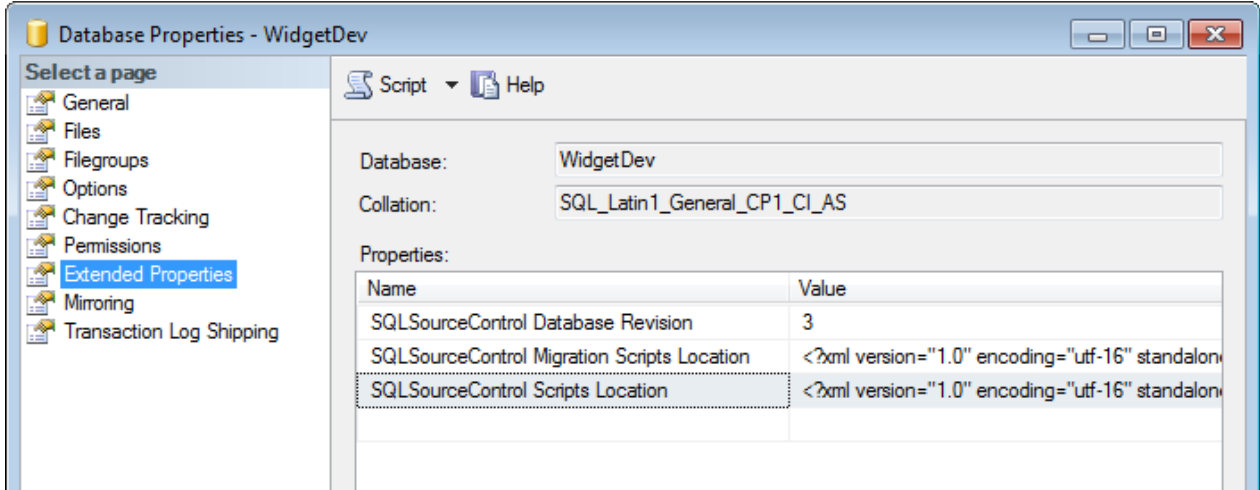
## /ScriptsFolderXML:<file path>

Alias: /sfx

The path to a text file that describes the location of a source control repository.

To create the text file:

1. In the SSMS Object Explorer, right-click a source-controlled database and click **Properties**.
2. In the Database Properties dialog box, click **Extended Properties**:



3. Copy the XML fragment from the *SQLSourceControl Scripts Location* extended property.
4. Create a new text file and paste the XML fragment into it.
5. Save the file.

## /Server1:<server name>

Alias: /s1

Specifies the server on which the source (/db1:) database is located. If an explicit path isn't specified, it defaults to *Local*.

### Example

```
sqlcompare /server1:Widget_Server\SQL2008 /db1:WidgetStaging /db2:WidgetProduction
```

## /Server2:<server name>

Alias: /s2

Specifies the server on which the target (/db2:) database is located. If an explicit path isn't specified, it defaults to *Local*.

## /ShowWarnings

Alias: /warn

Displays any warnings that apply to the deployment.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /showWarnings
```

For more information about warnings in SQL Compare, see [Warnings](#).

## **/Snapshot1:<file name>**

Alias: */sn1*

Specifies the snapshot to use as the source.

### **Example**

```
sqlcompare /snapshot1:"C:\Snapshots\WidgetStagingSnapshot.snp" /db2:WidgetProduction
```

## **/Snapshot2:<file path>**

Alias: */sn2*

Specifies the snapshot to use as the target.

### **Example**

```
sqlcompare /db1:WidgetStaging /snapshot2:"C:\Snapshots\WidgetProductionSnapshot.snp"
```

## **/Sourcecontrol1**

Allows a folder of scripts to be used as the source.

## **/Sourcecontrol2**

Allows a folder of scripts to be used as the target.

## **/Synchronize**

Aliases: */sync* or */synchronise*

Synchronizes (deploys) the databases after comparison.

The target (eg */db2*) is modified; the source (eg */db1*) isn't modified:

### **Example**

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /synchronize
```

## **/SyncScriptEncoding:<script encoding>**

Alias: */senc*

### **Arguments:**

<ul style="list-style-type: none"><li>• <i>UTF8</i></li></ul>	UTF-8 encoding, without preamble
<ul style="list-style-type: none"><li>• <i>UTF8WithPreamble</i></li></ul>	UTF-8 encoding, with 3-byte preamble
<ul style="list-style-type: none"><li>• <i>Unicode</i></li></ul>	UTF-16 encoding

- *ASCII*

ASCII encoding

Used with */scriptFile*. Specifies the character encoding used when writing the SQL script file. The default is UTF8.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /scriptFile:"C:\Scripts
Folder\WidgetSyncScript.sql" /syncScriptEncoding:ASCII
```

## **/Tempinstance:<tempinstance>**

Alias: */ti*

Specifies a connection string to a SQL Server instance to be used instead of LocalDB when generating deployment scripts that contain V2 migration scripts.

### Example

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /Tempinstance:"Data
Source=localhost;Integrated Security=SSPI;"
```

For more information about the temporary database, see [Setting the location of the temporary database](#) in the SQL Source Control documentation.

## **/TransactionIsolationLevel:<transaction isolation level>**

Alias: */til*

Specifies the Transaction Isolation Level to set in the SQL script.

## **/UserName1:<username>**

Alias: */u1*

The username for the source database.

If no username is specified, integrated security is used.

### Example

```
sqlcompare /db1:WidgetStaging /userName1:User1 /password1:P@ssw0rd
/db2:WidgetProduction /userName2:User2 /password2:Pa$$w0rd
```

## **/UserName2:<username>**

Alias: */u2*

The username for the target database.

If no username is specified, integrated security is used.

## **/Verbose**

Alias: */v*

Verbose mode.

### **/VersionUserName1:<username>**

Alias: */vu1*

Specifies the username for the source control server linked to the source database.

#### **Example**

```
sqlcompare /db1:WidgetStaging /v1:3 /versionUserName1:User1 /vp1:P@ssw0rd  
/db2:WidgetProduction /v2:HEAD /versionUserName2:User2 /vp2:Pa$$w0rd
```

If you have a username saved in SQL Source Control, you don't need to specify it in the command line.

### **/VersionUserName2:<username>**

Alias: */vu2*

Specifies the username for the source control server linked to the target database.

### **/VersionPassword1:<password>**

Alias: */vp1*

Specifies the password for the source control server linked to the source database.

#### **Example**

```
sqlcompare /db1:WidgetStaging /v1:3 /vu1:User1 /versionpassword1:P@ssw0rd  
/db2:WidgetProduction /v2:HEAD /vu2:User2 /versionpassword2:Pa$$w0rd
```

If you have a password saved in SQL Source Control, you don't need to specify it in the command line.

### **/VersionPassword2:<password>**

Alias: */vp2*

Specifies the password for the source control server linked to the target database.

## **Deprecated options**

### **/AllowIdenticalDatabases**

This switch is deprecated. Instead use */include:Identical*

*/include:Identical* suppresses the exit code if the two data sources are identical.

If */include:Identical* isn't set, and the data sources are identical, SQL Compare returns the error code 63.

### **/IncludeIdentical:<IncludeIdentical>**



This switch is deprecated. Instead use `/include:Identical`.

## `/MigrationsFolderXML:<file path>`

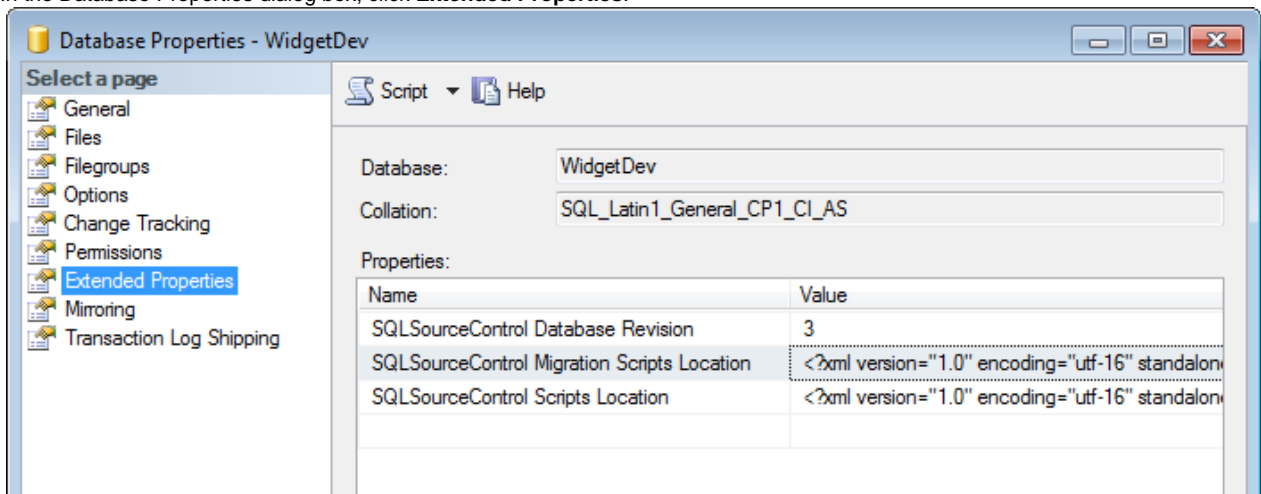
This switch is deprecated. Instead use `/migrationsfolder`.

Alias: `/mfx`

The path to a text file that describes the location of a migration scripts folder.

To create the text file:

1. In the SSMS Object Explorer, right-click a source-controlled database and click **Properties**.
2. In the Database Properties dialog box, click **Extended Properties**:



3. Copy the XML fragment from the `SQLSourceControl Migration Scripts Location` extended property.
4. Create a new text file and paste the XML fragment into it.
5. Save the file.

## Options used in the command line

You can set project configuration options by using the `/Options` switch.

For example, by default comparisons are not case-sensitive; to specify case-sensitive comparisons use:

```
/Options:CaseSensitiveObjectDefinition
```

To specify multiple options, separate the options using commas:

```
/Options:<option1>,<option2>,<option3>
```

If you don't explicitly set any options, the defaults are used.

### Default options:

- DecryptPost2KEncryptedObjects
- IgnoreFillFactor
- IgnoreWhiteSpace
- IncludeDependencies
- IgnoreFileGroups
- IgnoreUserProperties
- IgnoreWithElementOrder
- IgnoreDatabaseAndServerName

If you want to use these defaults with additional options, specify the *default* argument and the additional options. For example:

```
/Options:Default,CaseSensitiveObjectDefinition,IgnoreComments
```

If you don't specify the *default* argument, only the options you do specify apply.

To specify no options, use the *none* argument.

Further options are detailed below.

### AddDatabaseUseStatement

Alias: *adus*

Adds a USE statement at the top of the SQL deployment script.

### AddWithEncryption

Alias: *we*

Adds WITH ENCRYPTION when stored procedures, functions, views, and triggers are included in the deployment

If you use ADD ENCRYPTION on a SQL Server 2005 database, SQL Compare will not subsequently be able to compare or deploy the encrypted objects.

When SQL Compare creates a snapshot, this option is ignored, and WITH ENCRYPTION is not saved in the snapshot.

### CaseSensitiveObjectDefinition

#### Contents

- AddDatabaseUseStatement
- AddWithEncryption
- CaseSensitiveObjectDefinition
- ConsiderNextFilegroupInPartitionSchemes
- DecryptPost2KEncryptedObjects
- DisableAndReenableDdlTriggers
- DisableSOCForLiveDBs
- DoNotOutputCommentHeader
- DropAndCreateInsteadofAlter
- ForceColumnOrder
- IgnoreBindings
- IgnoreCertificatesAndCryptoKeys
- IgnoreChecks
- IgnoreCollations
- IgnoreComments
- IgnoreConstraintNames
- IgnoreDatabaseAndServerName
- IgnoreDataCompression
- IgnoreExtendedProperties
- IgnoreFileGroups
- IgnoreFillFactor
- IgnoreFullTextIndexing
- IgnoreIdentityProperties
- IgnoreIdentitySeedAndIncrement
- IgnoreIndexes
- IgnoreIndexLockProperties
- IgnoreInsteadOfTriggers
- IgnoreKeys
- IgnoreMigrationScripts
- IgnoreNotForReplication
- IgnorePermissions
- IgnoreQueueEventNotifications

Alias: *cs*

For databases with case-sensitive collation, enables objects with case-sensitive names to be compared and deployed. 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 or case-sensitive sort order.

Take care when you change this option. For example, if you create a schema snapshot with this option selected and you then compare the snapshot with another database without this option set, SQL Compare may produce unexpected errors.

## ConsiderNextFilegroupInPartitionSchemes

Alias: *cfgps*

This option isn't used for SQL Server 2000 databases.

When this option is selected, if a partition scheme contains a next filegroup, SQL Compare considers the next filegroup in the comparison and deployment if the partition scheme is extended. The next filegroup doesn't affect how data is stored.

## DecryptPost2KEncryptedObjects

Alias: *dp2k*

This option isn't used for SQL Server 2000 databases.

When this option is specified, SQL Compare decrypts text objects in databases created using the WITH ENCRYPTION option.

- When comparing large databases with few encrypted objects, selecting this option may result in slower performance.
- When this option isn't selected, text objects are shown as different and can't be deployed.

## DisableAndReenableDdlTriggers

Alias: *drd*

This option isn't used for SQL Server 2000 databases.

DDL triggers can cause problems when you run the deployment. Select this option to disable any enabled DDL triggers before deploying the databases, and re-enable those triggers following deployment.

## DisableSOCForLiveDBs

Alias: *dafld*

When this option is specified, SQL Compare won't retrieve migration scripts when you compare a database. (By default, when you compare a database that has an associated revision number, SQL Compare tries to connect to source control to retrieve any relevant migration scripts.) This option can be useful if there's a problem connecting to source control when comparing a database.

## DoNotOutputCommentHeader

Alias: *nc*

When this option is specified, comments and comment headers aren't included in the output deployment script.

## DropAndCreateInsteadofAlter

Alias: *dacia*

- IgnoreQuotedIdentifiersAndANSINullSettings
- IgnoreReplicationTriggers
- IgnoreSchemaObjectAuthorization
- IgnoreSquareBrackets
- IgnoreStatistics
- IgnoreStatisticsNorecompute
- IgnoreSystemNamedConstraintNames
- IgnoreTriggerOrder
- IgnoreTriggers
- IgnoreTSQLt
- IgnoreUserProperties
- IgnoreUsersPermissionsAndRoleMemberships
- IgnoreWhiteSpace
- IgnoreWithElementOrder
- IgnoreWithEncryption
- IgnoreWithNocheck
- IncludeDependencies
- none
- NoTransactions
- ObjectExistenceChecks
- ThrowOnFileParseFailed
- UseClrUdtToStringForClrMigration

When this option is specified, SQL Compare replaces ALTER statements in the deployment script with DROP and CREATE statements for the following objects:

- Views
- Stored Procedures
- Functions
- Extended Properties
- DDL Triggers
- DML Triggers

If you specify this option, you must also select the *Add Object Existence Checks* option, or the deployment script will fail.

## ForceColumnOrder

Alias: *f*

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 deployment. Data will be preserved.

## IgnoreBindings

Alias: *ib*

Ignores bindings on columns and user-defined types when comparing and deploying (eg *sp\_bindrule* and *sp\_bindefault* clauses would be ignored).

## IgnoreCertificatesAndCryptoKeys

Alias: *icc*

This option is used only for SQL Server 2005 databases.

SQL Server severely restricts access to certificates, symmetric keys, and asymmetric keys. This means SQL Compare can't compare all of the properties for a symmetric key.

If certificates, symmetric keys, and asymmetric keys are selected for deployment, only the permissions are deployed.

## IgnoreChecks

Alias: *ich*

Ignores check constraints when comparing and deploying databases.

## IgnoreCollations

Alias: *ic*

Ignores collation orders on character datatype columns when comparing and deploying databases.

## IgnoreComments

Alias: *icm*

Ignores comments when comparing views, stored procedures and so on. Comments will still appear in the deployment scripts.

## IgnoreConstraintNames

Alias: *icn*

Ignores the names of indexes, foreign keys, primary keys, and default, unique, and check constraints when comparing databases. Names won't be ignored when the databases are deployed.

## IgnoreDatabaseAndServerName

Ignores the names of databases and servers when comparing databases.

This option isn't used for SQL Server 2000 databases.

## IgnoreDataCompression

Alias: *idc*

Ignores data compression on indexes and tables.

## IgnoreExtendedProperties

Alias: *ie*

Ignores extended properties on objects and databases when comparing and deploying databases.

## IgnoreFileGroups

Alias: *ifg*

Ignores filegroup clauses, partition schemes, and partition functions on tables and keys when comparing and deploying databases. Partition schemes and partition functions aren't displayed in the comparison results.

## IgnoreFillFactor

Alias: *if*

Ignores the fill factor and index padding in indexes and primary keys when comparing and deploying databases.

## IgnoreFullTextIndexing

Alias: *ift*

Ignores full-text catalogs and full-text indexes when comparing and deploying databases.

## IgnoreIdentityProperties

Alias: *iip*

Ignores the identity property on columns when comparing databases. The identity property won't be ignored when databases are deployed.

## IgnoreIdentitySeedAndIncrement

Alias: *isi*

For identity properties, ignores only the identity seed and increment values when comparing databases. They won't be ignored when the databases are deployed.

## IgnoreIndexes

Alias: *ii*

Ignores indexes, statistics, unique constraints, and primary keys when comparing and deploying databases.

## IgnoreIndexLockProperties

Alias: *iilp*

Ignores the lock properties of indexes.

## IgnoreInsteadOfTriggers

Alias: *iit*

Ignores INSTEAD OF DML triggers when comparing and deploying databases.

## IgnoreKeys

Alias: *ik*

Ignores foreign keys when comparing and deploying databases.

## IgnoreMigrationScripts

Alias: *ims*

Ignores migration scripts when comparing and deploying databases. For information about migration scripts, see [Working with migration scripts](#) (SQL Source Control documentation).

## IgnoreNotForReplication

Alias: *infr*

Ignores the NOT FOR REPLICATION option on foreign keys, identities, check constraints and triggers.

If you specify this option, the NOT FOR REPLICATION statement won't be displayed in the object creation script for foreign keys, identities, and check constraints.

For triggers, the NOT FOR REPLICATION statement will be displayed in the object creation script, but will be ignored for the purposes of the comparison. When comparing triggers, you should also specify the **Ignore white space** option, but this option will also be applied to all objects in the comparison.

Check constraints and foreign keys that contain the NOT FOR REPLICATION statement in their definition will automatically be flagged as WITH NOCHECK. Use the **Ignore WITH NOCHECK** option to identify these objects as being the same.

## IgnorePermissions

Alias: *ip*

Ignores permissions on objects when comparing and deploying databases.

## IgnoreQueueEventNotifications

Alias: *iqen*

This option isn't used for SQL Server 2000 databases.

Ignores the event notification on queues when comparing and deploying databases.

## IgnoreQuotedIdentifiersAndAnsiNullSettings

Alias: *iq*

Ignores SET QUOTED\_IDENTIFIER and SET ANSI\_NULLS statements. Ignores these common SET statements when comparing views, stored procedures and so on. These statements won't be ignored when the databases are deployed.

## IgnoreReplicationTriggers

Alias: *irpt*

Ignores replication triggers when comparing and deploying databases.

## **IgnoreSchemaObjectAuthorization**

Alias: *isoa*

Ignores authorization clauses on schema objects.

## **IgnoreSquareBrackets**

Alias: *isb*

Ignores starting and ending square brackets in object names which have been escaped using square brackets. This applies to textual objects such as stored procedures, triggers, etc.

## **IgnoreStatistics**

Alias: *ist*

Ignores statistics when comparing and deploying databases.

## **IgnoreStatisticsNorecompute**

Alias: *isn*

Ignores STATISTICS\_NORECOMPUTE on indexes.

## **IgnoreSystemNamedConstraintNames**

Alias: *iscn*

Ignores system named constraint and index names. Ignores the names of system named indexes, foreign keys, primary keys, default, unique and check constraints. The names will still be scripted and deployed.

## **IgnoreTriggerOrder**

Alias: *ito*

DML triggers can have an order specified, such as FIRST INSERT, LAST UPDATE, and so on. Specify this option to ignore the trigger order for DML triggers when comparing and deploying databases. The DDL trigger order isn't affected.

## **IgnoreTriggers**

Alias: *it*

Ignores DML triggers when comparing and deploying databases.

## **IgnoretSQLt**

Alias: *itst*

Ignores the tSQLt schema and its contents, the tSQLtCLR assembly, the SQLCop schema and its contents, and any schemas and their contents with the tSQLt.TestClass extended property set.

## **IgnoreUserProperties**

Alias: *iup*

This option isn't used for SQL Server 2000 databases.

If you specify this option, users' properties are ignored, and only the user name is compared and deployed.

If you don't specify this option, SQL Compare compares user properties, such as the type of user (SQL, Windows, certificate-based, asymmetric key based) and any schema. If a user is selected for deployment, SQL Compare deploys the properties where possible.

## IgnoreUsersPermissionsAndRoleMemberships

Alias: *iu*

Ignores users' permissions and role memberships.

## IgnoreWhiteSpace

Alias: *iw*

Ignores white space (newlines, tabs, spaces, and so on) when comparing databases. White space won't be ignored when the databases are deployed.

## IgnoreWithElementOrder

Alias: *iweo*

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), specify this option to ignore the order of the WITH elements when comparing and deploying databases.

## IgnoreWithEncryption

Alias: *iwe*

Ignores WITH ENCRYPTION statements on triggers, views, stored procedures and functions. This option overrides Add WITH ENCRYPTION.

## IgnoreWithNocheck

Alias: *iwn*

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

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

## IncludeDependencies

Alias: *incd*

Includes dependent objects when comparing and deploying databases. For example, if a view depends on a table then the table will be deployed when deploying the view.

## none

Alias: *n*

To specify no options, use the *none* argument.

## NoTransactions

Alias: *nt*

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

If this option isn't specified and the deployment script fails, the script is rolled back to the start of the failed transaction. If this option is specified, the script isn't rolled back. This can be useful for detection of errors within a script.



## **ObjectExistenceChecks**

Alias: *oec*

Checks for the existence of objects affected by the deployment by adding IF EXISTS statements in the deployment script.

This option can be useful if you want to run the deployment script multiple times.

## **ThrowOnFileParseFailed**

Alias: *tofpf*

Throws an exception when parsing a scripts folder fails.

## **UseClrUdtToStringForClrMigration**

Alias: *uclr*

This option isn't used for SQL Server 2000 databases.

If CLR objects included in the deployment, this option forces two rebuilds of the table with conversion to and from strings to update the CLR objects, instead of using ALTER ASSEMBLY. For more information, see [Understanding the deployment](#).

This option affects the deployment only.

## Exit codes used in the command line

If a task you are performing with the SQL Compare command line interface fails, and you do not see an error message explaining the reason for the failure, you may see one of the exit codes listed below:

### 0 - Success

### 1 - General error code

### 3 - Illegal argument duplication

Some arguments must not appear more than once in a command line.

### 8 - Unsatisfied argument dependency

There is an unsatisfied argument dependency or violated exclusion when the command line is run. For example:

- */arg2* depends on */arg1* but you have specified */arg2* without specifying */arg1*
- */arg2* can't be used with */arg1* but you have used both

### 32 - Value out of range

The numeric value supplied for an argument is outside the range of valid values for that argument.

### 33 - Value overflow

The value supplied for an argument is too large.

### 34 - Invalid value

The value supplied for an argument is invalid.

### 35 - Invalid license

Software license or trial period has expired.

### 61 - Deployment warnings

SQL Compare encountered serious warnings that apply to the deployment.

If you're using SQL Compare, you can ignore these warnings by specifying */AbortOnWarnings:None*

If you're using SQLCI, you can ignore these warnings by specifying */additionalCompareArgs=/AbortOnWarnings:None*

For more information about */AbortOnWarnings*, see [Switches used in the command line](#).

### 62 - High level parser error

SQL Compare encountered high level errors when parsing a scripts folder.

Use */IgnoreParserErrors* to force SQL Compare to continue without exiting.

The **Error Parsing Scripts** dialog box in the SQL Compare user interface provides additional information to help you resolve script parser errors.

### 63 - Databases identical

The databases being compared are identical or no objects have been included.

To suppress this error use */Include:Identical*

#### **64 - Command line usage error**

The command line was used incorrectly. For example, an incorrect flag, or incorrect syntax may have been used.

#### **65 - Data error**

Data required by SQL Compare is invalid or corrupt.

#### **69 - Resource unavailable**

A resource or service required to run SQL Compare is unavailable.

#### **70 - An unhandled exception occurred**

See the log for more details.

#### **73 - Failed to create report**

The report was not created.

#### **74 - I/O error**

This is returned if SQL Compare attempts to write to a file that already exists, and the */force* switch has not been set.

#### **77 - Insufficient permission**

The action can't be completed because the user does not have the necessary permission.

#### **79 - Databases not identical**

This is returned when the */assertidentical* switch is used and the source and target are not identical.

#### **126 - SQL Server error**

Execution failed because of an error.

#### **130 - Ctrl-Break**

Execution stopped because of a Ctrl-Break.

#### **400 - Bad request**

The command line arguments can't be executed. For example, you may have provided two mutually exclusive switches.

#### **402 - Not licensed**

There is no acceptable license. If you have one, use */activateSerial:<SerialKey>*. If you don't have a license, please contact [sales@red-gate.com](mailto:sales@red-gate.com) for a either a trial extension key or to purchase a license.

#### **499 - Activation cancelled by user**

Activation was cancelled because the cancel button was pressed during the process.

## **500 - Unhandled exception**

An unhandled exception occurred. The exception message and stack trace will be included in the output.

# Getting more from SQL Compare

These pages explain some advanced features of SQL Compare.

- [Comparing databases on different SQL Server versions](#)
- [Deploying to SQL Azure](#)
- [Creating a rollback script](#)
- [Logging and log files](#)
- [Forcing SQL Compare and SQL Data Compare to use an encrypted connection](#)
- [Copying the structure of a database](#)

## Comparing databases on different SQL Server versions

This topic provides additional information if you're comparing databases on different versions of Microsoft SQL Server.

### Comparing a SQL Server 2005 or 2008 database with a SQL Server 2000 database

If you're updating a SQL Server 2008 or SQL Server 2005 database to match a SQL Server 2000 database, you must not change the default [project options](#) on the Project Configuration dialog box.

However, if your database is on a SQL Server with case-sensitive sort order, you must select the **Treat items as case sensitive** [project option](#).

If you're updating a SQL Server 2000 database to match a SQL Server 2005 database, note that:

- SQL Compare may be unable to deploy all objects. [Deployment warnings](#) are displayed where possible.
- SQL Compare version 7.0 (and earlier) can't decrypt objects that are encrypted in a SQL Server 2005 or SQL Server 2008 database.  
If an encrypted object which can't be decrypted exists in both databases, it is shown under the **objects that exist in both but are different** group in the upper (Results) pane. SQL Compare can't compare the encrypted objects, or display their creation scripts, and can't deploy them.
- In SQL Compare version 7.1 (and later) you have the option to decrypt text objects in SQL Server 2008 and SQL Server 2005 databases that were created using the WITH ENCRYPTION option.  
Disabling this option can result in faster performance. To disable this option, on the **Options** tab of the Project Configuration dialog, clear the **Decrypt encrypted objects on 2005 and 2008 databases** check box.  
When this option is disabled, SQL Compare can't compare the encrypted objects, or display their creation scripts, and can't deploy them.
- An extra line of white space may be appended to the creation scripts of stored procedures, functions, rules, and so on. So that SQL Compare does not flag these objects as different, you should select the project option [Ignore white space](#).

When you create a default value or constraint in SQL Server 2005, the definitions of the default value or constraint are parsed by SQL Server 2005, and the parsed version is stored. The syntax of the 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))**. SQL Compare highlights these differences in the lower (SQL Differences) pane, and if these are the only differences, it shows the objects to be identical.

For more information, see: [Viewing the SQL differences](#).

### Comparing SQL Server 2005 compatibility level 80 databases

If a 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, comparisons and deployments may fail.

### Comparing SQL Server 2005 compatibility level 90 databases

If a SQL Server 2008 database has its compatibility level set to 90, it conforms to strict rules for views, stored procedures, functions, and DML triggers. Therefore, comparisons and deployments may fail.

## Deploying to SQL Azure

With SQL Compare, you can compare and deploy SQL Azure databases.


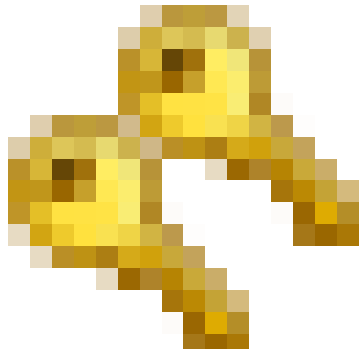

### Object types not supported

SQL Azure databases do not support the following object types:

	Application Roles		Assemblies
	Asymmetric Keys		Certificates
	Contracts		Defaults
	Event Notifications		Full Text Catalogs

	Message Types		Partition Functions
	Partition Schemes		Queues
	Application Roles		Routes
	Rules		Services



	Service Bindings		Extended Stored Procedures
	Numbered Stored Procedures		Symmetric Keys
	Remote Synonyms		System Tables
	User-Defined Types		XML Schema Collections

If you try to deploy an unsupported object to a SQL Azure database, the deployment script will fail.

## T-SQL and SQL Server feature support

SQL Azure has further limitations that can cause SQL Compare deployments to fail. For example, SQL Azure does not support Encryption, Data Compression, or SQL Server Replication.

For full details of these limitations, refer to your SQL Azure documentation:

- [T-SQL Support \(SQL Azure\)](#)
- [SQL Server Feature Limitations \(SQL Azure\)](#)

## Creating a rollback script

If you want to reverse a deployment, or return a database to a specific state, you can create a rollback script.

Before you run the deployment wizard:

1. In the main window, right click in the Direction Bar, and click **Switch deployment direction**.
2. Use the deployment wizard to create and save a deployment script.

For example, if you are migrating changes from *WidgetStaging* (the source) to *WidgetProduction* (the target), switching the deployment direction makes *WidgetProduction* the source, and the deployment script you create can be run on *WidgetProduction* in future to restore it to its current state.

You can also roll back a deployment by creating a snapshot when you deploy:

On the first page of the deployment wizard, select the **Back up target before deployment** check box. This adds a page to the wizard where you can choose to create a schema snapshot from the target. The snapshot preserves the state of the target before the deployment, and later can be used as a source, to restore the target to that state.

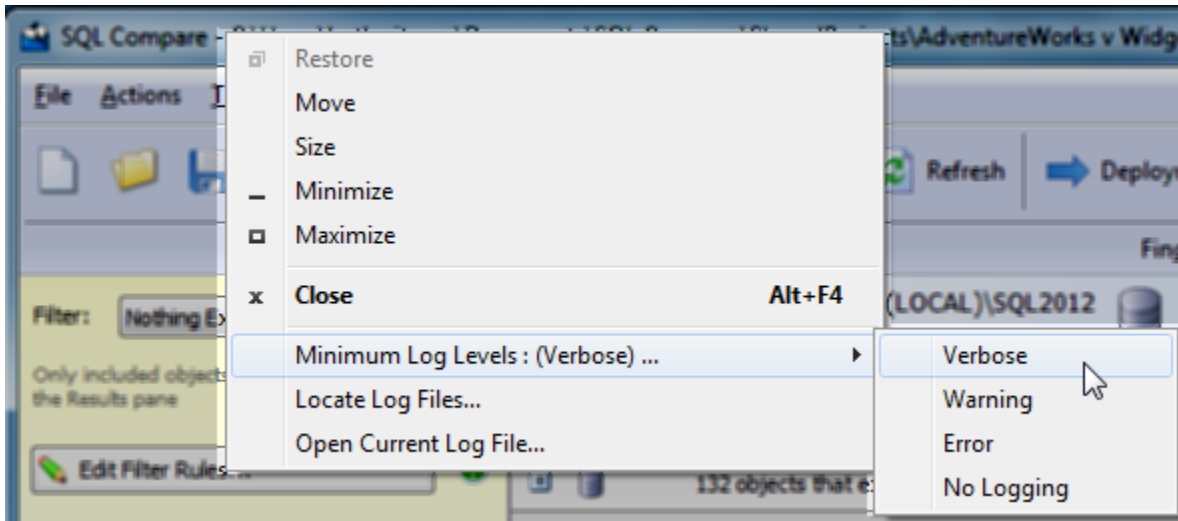
## Logging and log files

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, select the log level you require:

- Right-click the application title bar, click **Minimum Log Levels**, and then click the required log level.



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

<b>Verbose</b>	Reports all messages in the log file.
<b>Warning</b>	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.
<b>Error</b>	Reports serious and fatal errors. For example, an error message might report a failed operation.
<b>No Logging</b>	Disables logging.

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.

## Forcing SQL Compare and SQL Data Compare to use an encrypted connection

By default, SQL Compare and SQL Data Compare don't have an option to force an encrypted connection when connecting to live databases.

To force an encrypted connection, you can add the encryption properties into the SQL Compare or SQL Data Compare connection string:

1. Open SQL Compare or SQL Data Compare.
2. Create a new project or edit an existing project.
3. In the **Data Sources** tab, go to the **Database** field.
4. Type or paste one of the following:
  - a. For a default instance:

```
<Server Name>;ENCRYPT=TRUE;TRUSTSERVERCERTIFICATE=TRUE
```

- b. For a named instance:

```
<Server Name>\<Instance Name>;ENCRYPT=TRUE;TRUSTSERVERCERTIFICATE=TRUE
```

- c. For a default instance not running on the default SQL port (1433):

```
<Server Name>,<port number>;ENCRYPT=TRUE;TRUSTSERVERCERTIFICATE=TRUE
```

- d. For a named instance not running on the default SQL port (1433):

```
<Server Name>\<Instance Name>,<port  
number>;ENCRYPT=TRUE;TRUSTSERVERCERTIFICATE=TRUE
```

SQL Compare or SQL Data Compare will now use an encrypted connection.

## Copying the structure of a database

You can use SQL Compare to copy the structure of an existing database to a new database.

To copy the structure of a database:

1. Use your SQL application to create the new database.
2. Create a project that compares the existing database, scripts folder, backup or SQL Compare snapshot with the new database.
3. Run the comparison on the project.
4. Select the database objects that you want to copy.
5. Deploy the databases.

Alternatively, Red Gate offers **SQL Packager**, which will script the structure, and optionally the contents of a database. You can create a .NET executable file or C# project, which will enable you to create the entire database.

## Worked examples

These detailed examples demonstrate how to complete tasks using SQL Compare.

- [Worked example - comparing and deploying two databases](#)
- [Worked example - using a scripts folder as a data source](#)

# Worked example - comparing and deploying two databases

This worked example demonstrates a basic comparison and deployment of two SQL Server databases.

In the example, the Magic Widget Company has a SQL Server database running on a live web server. This database contains a number of tables, views, stored procedures, and other database objects. The Magic Widget Company's development team has been working on an upgrade to their website. As part of this upgrade, they have made a number of changes to the structure of the database. They have already deployed the changes from the development server to a staging server.

They now need to deploy the changes to the production server.

This example has four steps:

1. [Set up the databases](#)  
Create the example databases on your SQL Server.
2. [Set up the comparison](#)  
Specify the data sources you want to compare.
3. [Select objects to deploy](#)  
Review the results and select the objects you want to deploy.
4. [Deploy the databases](#)  
Create and run a deployment script.

## 1. Set up the databases

The worked example uses the following databases:

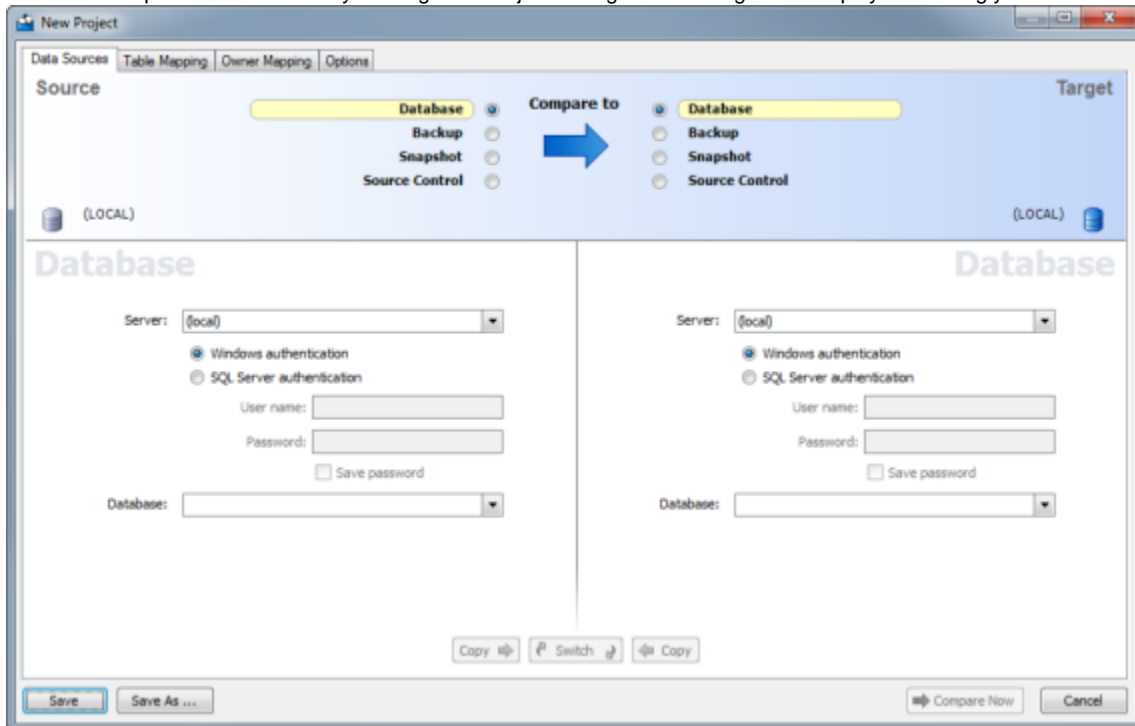
- WidgetStaging is the staging database
- WidgetProduction is the production database

To create these two databases on your SQL Server:

1. If they already exist, delete the databases *WidgetStaging* and *WidgetProduction*.
2. [Click here to download](#) the SQL creation script for the databases.
3. Copy the script, paste it in your SQL editor application, and run it.  
The databases and their schema are created.

## 2. Set up the comparison

1. Start SQL Compare if it is not already running. The Project Configuration dialog box is displayed showing your most recent project:



You can edit the current project, or create a new project.

If you want to create a new project, click **Cancel** to close the dialog box, and on the toolbar click

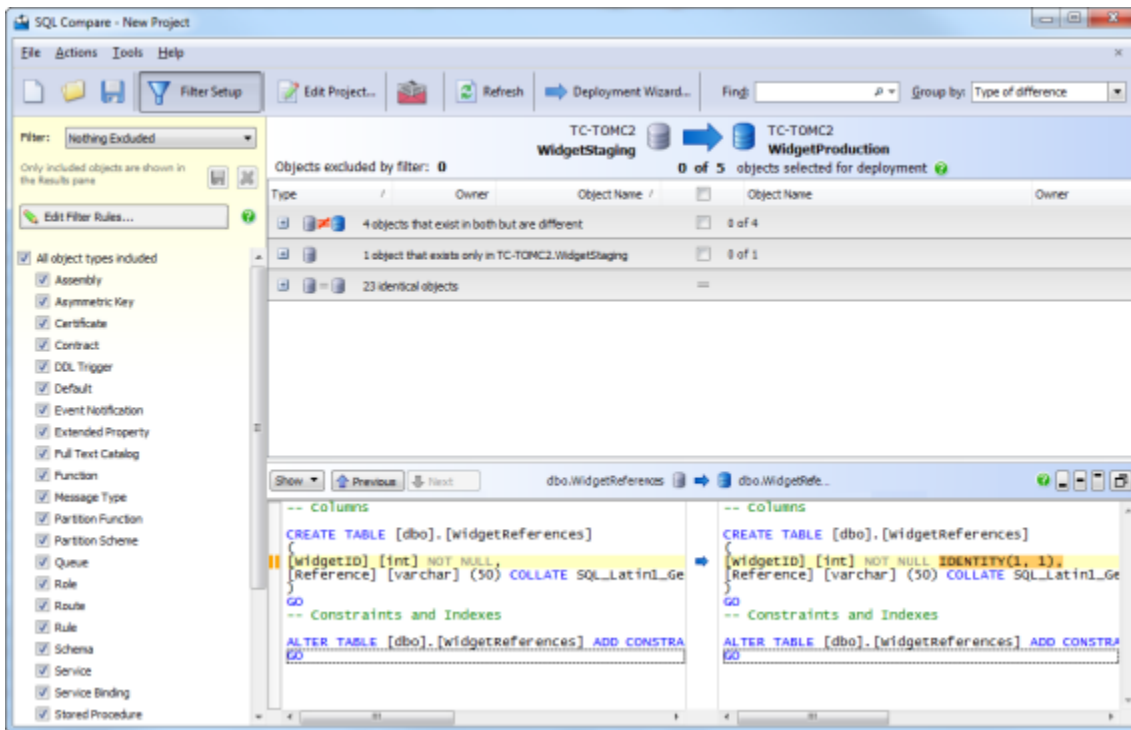


(New Project)

- In the shaded upper pane, ensure both Source and Target are set to **Database**.  
In this example, we will compare databases. You can also compare backups, SQL Compare snapshots, and scripts folders.
- For each data source, in **Server**, type or select the name of the server on which you set up the databases.
- For the source, in **Database**, type or select *WidgetStaging*. Type or select *WidgetProduction* for the target.  
If the databases are not displayed in the **Database** lists, right-click in each **Database** box and click **Refresh**, or scroll to the top of the list and click **Refresh**.
- Click **Compare Now**.  
SQL Compare displays a message dialog box that shows the progress of the comparison.  
When the comparison is complete, click **OK** to close the message box.

### 3. Select objects to deploy

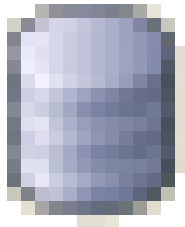
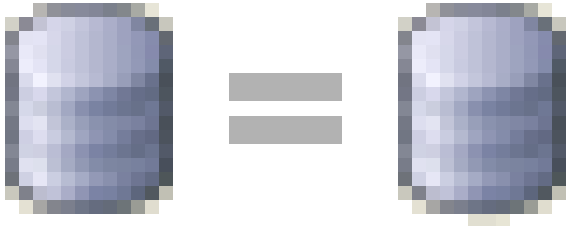
The results of the comparison are displayed in the upper (Results) pane:



The results are grouped by:





	objects that exist in WidgetStaging but do not exist in WidgetProduction
	objects that exist in both databases and are identical

To view the objects in a group, click

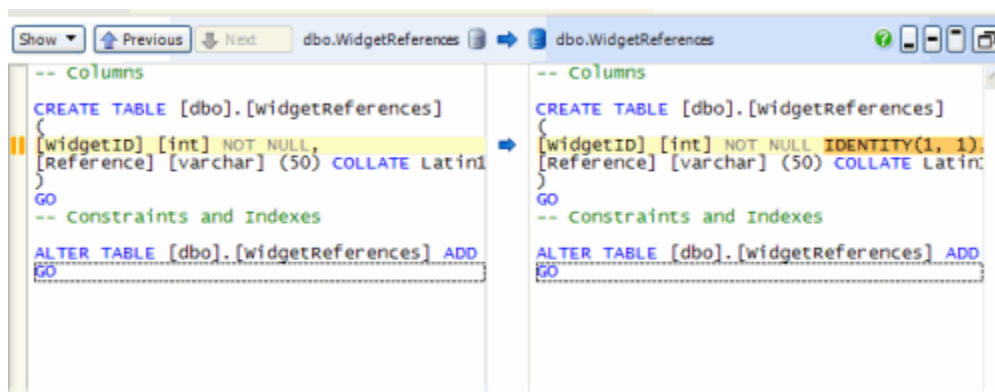


:

Type	Object Name	<input type="checkbox"/>	Object Name
4 objects that exist in both but are different		<input type="checkbox"/>	0 of 4
Table	WidgetPrices	<input type="checkbox"/>	WidgetPrices
Table	WidgetReferences	<input type="checkbox"/>	WidgetReferences
Table	Widgets	<input type="checkbox"/>	Widgets
View	CurrentPrices	<input type="checkbox"/>	CurrentPrices
1 object that exists only in (local).WidgetStaging		<input type="checkbox"/>	0 of 1
10 identical objects		<input type="checkbox"/>	0 of 10

When you click an object, the lower (SQL Differences) pane shows a side-by-side, color-coded listing of the differences in the object creation scripts.

This example shows the *WidgetReferences* table:



```

-- Columns
CREATE TABLE [dbo].[widgetReferences]
(
[widgetID] [int] NOT NULL,
[Reference] [varchar] (50) COLLATE Latin1
)
GO
-- Constraints and Indexes
ALTER TABLE [dbo].[widgetReferences] ADD
GO

-- Columns
CREATE TABLE [dbo].[widgetReferences]
(
[widgetID] [int] NOT NULL IDENTITY(1, 1),
[Reference] [varchar] (50) COLLATE Latin1
)
GO
-- Constraints and Indexes
ALTER TABLE [dbo].[widgetReferences] ADD
GO

```

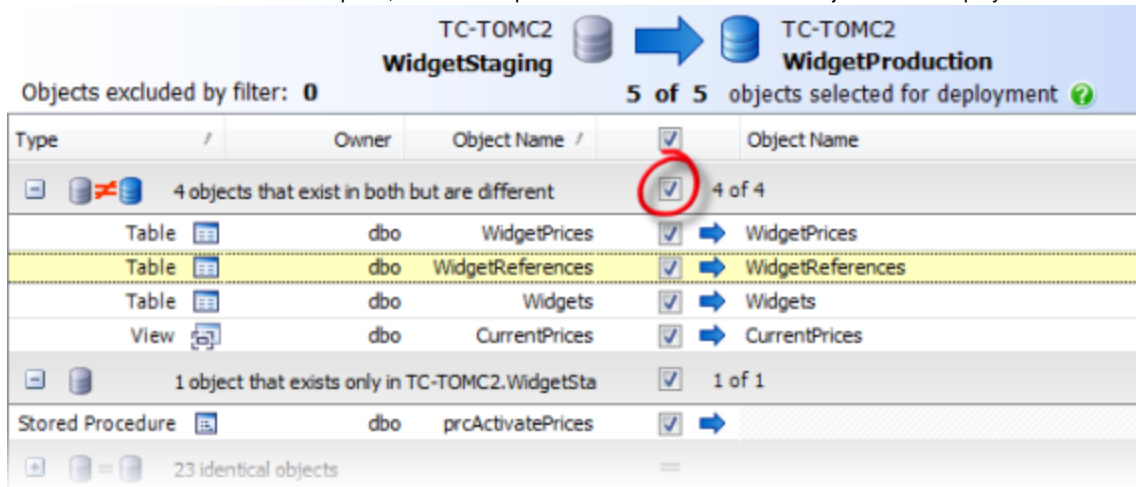
For more information, see:

- Viewing the comparison results
- Viewing the SQL differences

Use the check boxes in the middle of the upper (Results) pane to select objects for deployment.

In this example, we will deploy all objects that are different:

- On the Filter pane, ensure the default filter *Nothing Excluded* is selected.  
The current filter defines which objects are displayed. When you use the filter to exclude an object, it is removed from the Results pane and can't be selected for deployment.  
For more information, see: [Using filters](#).
- In the central column of the Results pane, select the top level check box to include all objects in the deployment:



All objects are selected.

- Click



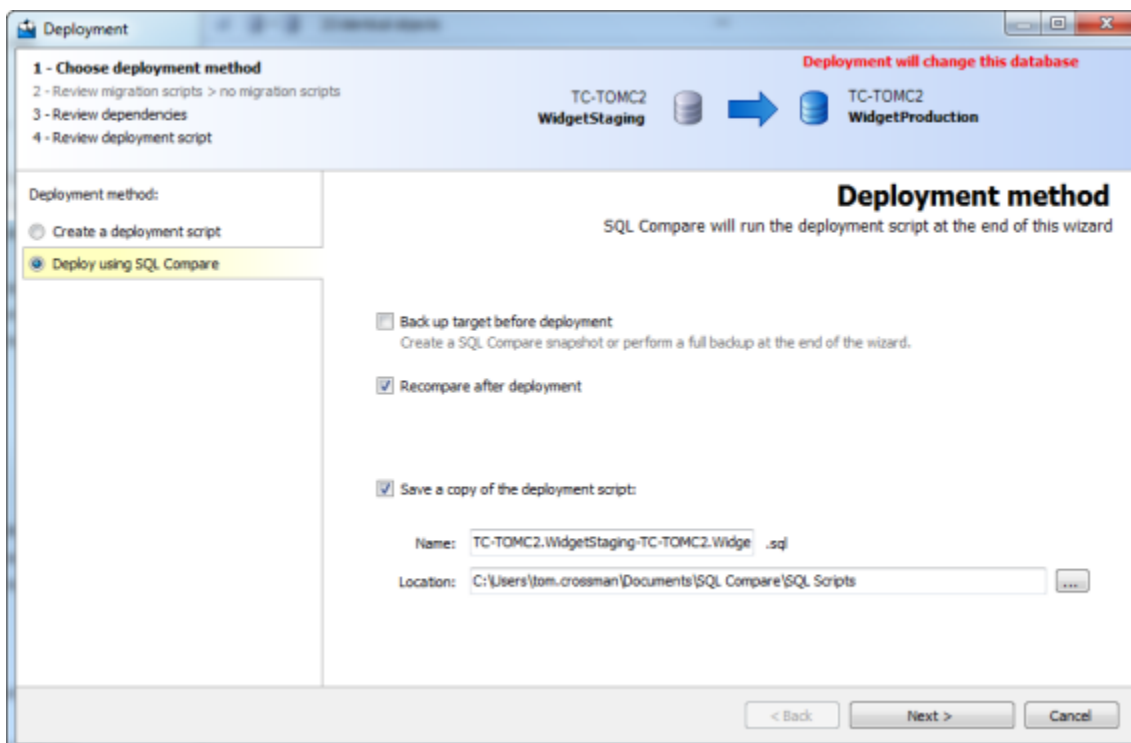
**Deployment Wizard.**

#### 4. Deploy the databases

On the first page of the deployment wizard you can choose to create and save a deployment script, or perform the deployment using SQL Compare.

#### Choose deployment method

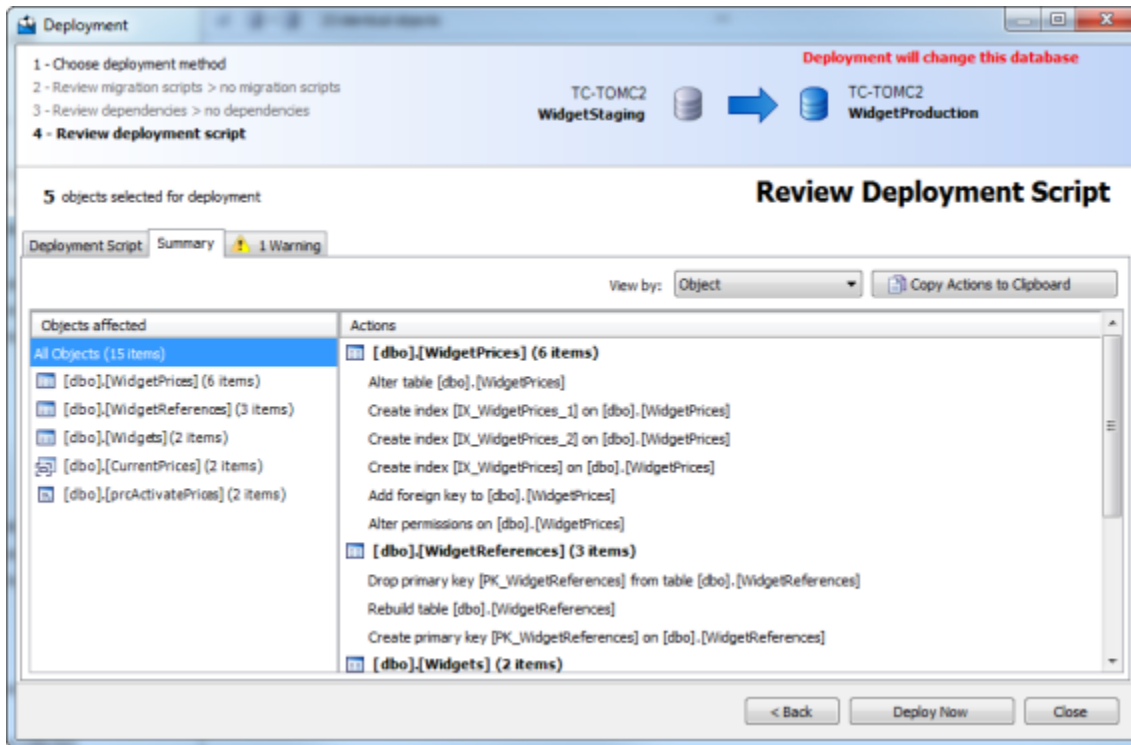
In this example, we will deploy using SQL Compare.



1. Ensure that **Deploy using SQL Compare** is selected.
2. Clear the **Back up target before deployment** check box if it is selected.  
In this example, we will not back up before deployment.  
For more information, see: [Backing up before deployment](#).
3. Ensure the **Recompare after deployment** check box, and the **Save a copy of the deployment script** check box are selected.  
In this example, we will run the script from SQL Compare, and compare the databases afterwards to check the results. We will also save a copy of the deployment script.
4. Click **Next**.

## Review script

The final page of the wizard displays information about the deployment:



There are three tabs on the **Review** page:

- **Deployment script** shows the script to deploy the data sources.  
You can search the script, save it, or copy it to the clipboard.
- **Summary** shows a synopsis of the actions in the deployment script.  
You can view the summary grouped by the objects affected, by the type of modification, or by the order in which the script modifies the target.
- **Warnings** shows a list of any warnings about unexpected behavior that may occur when you deploy the databases.  
For more information, see: [Deployment warnings](#).

In this example, SQL Compare displays a warning to inform you that it can't use the ALTER TABLE command to change the IDENTITY column, so the deployment script will rebuild the *WidgetReferences* table.

Warnings are displayed whenever tables require rebuilding as these may be slow operations. Data in tables is preserved when tables are rebuilt.

## Performing the deployment

When you have reviewed the script, deploy the databases:

- Click **Deploy Now** to perform the deployment.
- A confirmation dialog box is displayed. Click **Deploy Now** to continue.
- SQL Compare displays a message dialog box that shows the progress of the deployment.  
When the deployment is complete, click **OK** to close the message box.

SQL Compare then re-compares the databases. The results are shown in the main window. In this example, all objects are shown to be identical, confirming that the deployment has been a success:

SQL Compare - New Project

File Actions Tools Help

Filter Setup Edit Project... Refresh Deployment Wizard... Find: [ ] Group by: Type of difference

TC-TOMC2 WidgetStaging TC-TOMC2 WidgetProduction

Objects excluded by filter: 0 All objects identical

28 identical objects

Type	Owner	Object Name /	Object Name	Owner
Table	dbo	WidgePrices	WidgePrices	dbo
Table	dbo	WidgeReferences	WidgeReferences	dbo
Table	dbo	Widges	Widges	dbo
View	dbo	CurrentPrices	CurrentPrices	dbo
Stored Procedure	dbo	prActivatePrices	prActivatePrices	dbo
Role		db_accessadmin	db_accessadmin	
Role		db_backupoperator	db_backupoperator	
Role		db_datareader	db_datareader	
Role		db_datawriter	db_datawriter	
Role		db_ddadmin	db_ddadmin	
Role		db_denydatareader	db_denydatareader	
Role		db_denydatawriter	db_denydatawriter	
Role		db_owner	db_owner	
Role		db_securityadmin	db_securityadmin	
Role		public	public	
Schema		db_accessadmin	db_accessadmin	
Schema		db_backupoperator	db_backupoperator	
Schema		db_datareader	db_datareader	
Schema		db_datawriter	db_datawriter	
Schema		db_ddadmin	db_ddadmin	
Schema		db_denydatareader	db_denydatareader	
Schema		db_denydatawriter	db_denydatawriter	
Schema		db_owner	db_owner	
Schema		db_securityadmin	db_securityadmin	
Schema		dbo	dbo	

Filter: Nothing Excluded

Only included objects are shown in the Results pane

Edit Filter Rules...

- All object types included
- Assembly
- Asymmetric Key
- Certificate
- Contract
- DDL Trigger
- Default
- Event Notification
- Extended Property
- Full Text Catalog
- Function
- Message Type
- Partition Function
- Partition Scheme
- Queue
- Role
- Route
- Rule
- Schema
- Service
- Service Binding
- Stored Procedure

## Worked example - using a scripts folder as a data source

This worked example demonstrates the use of a scripts folder when comparing and deploying databases. A scripts folder contains SQL script files representing a database's structure and, optionally, data.

For more information, see: [Working with scripts folders](#).

In the example, the Super Sprocket Company has a SQL Server database that contains a number of tables, views, stored procedures, and other database objects. The Super Sprocket Company's development team has been given the task of making a number of changes to the structure of the database and updating the production server.

A copy of the production database has already been restored to an empty database, ready for development. To ensure no untested changes are made to the production database, they will also save its schema as a scripts folder.

You can follow the example on your own system, if you are using SQL Compare Professional edition. You will need access to a SQL Server to do this.

If you have not already followed the [Comparing and deploying two databases](#) worked example, you are recommended to do so before starting this worked example.

This example has four steps:

1. [Set up the databases](#)  
Create the example databases on your SQL Server.
2. [Set up the comparison](#)  
Specify the data sources you want to compare.
3. [Select objects to deploy](#)  
Review the results and select the objects you want to deploy.
4. [Create a deployment script](#)  
Create a script to update the production database.

You can only compare scripts folders if you are using SQL Compare Professional Edition.

### 1. Set up the databases

The worked example uses the following databases:

- *SprocketProduction* is the production database
- *SprocketDevelopment* is the modified version of the database containing the updates

To create these databases on your SQL Server:

1. If they already exist, delete the databases **SprocketProduction**, and **SprocketDevelopment** from your SQL Server.
2. [Click here to download the SQL creation script](#) for the databases.
3. Copy the script, paste it in your SQL editor, and run it.  
The databases and their schema are created.

### 2. Set up the comparison

Development on the copy of the production database proceeds, and at some point a milestone is reached; the next version is ready to be tested. The development team compares the modified database with the production database schema. To ensure no untested changes are made to the production database, they save its schema as a scripts folder.

Creating a scripts folder before deployment preserves the current state of the production database. It can be checked into source control, providing history and allowing you to revert any changes.

To do this, set up a comparison project:

1. Click

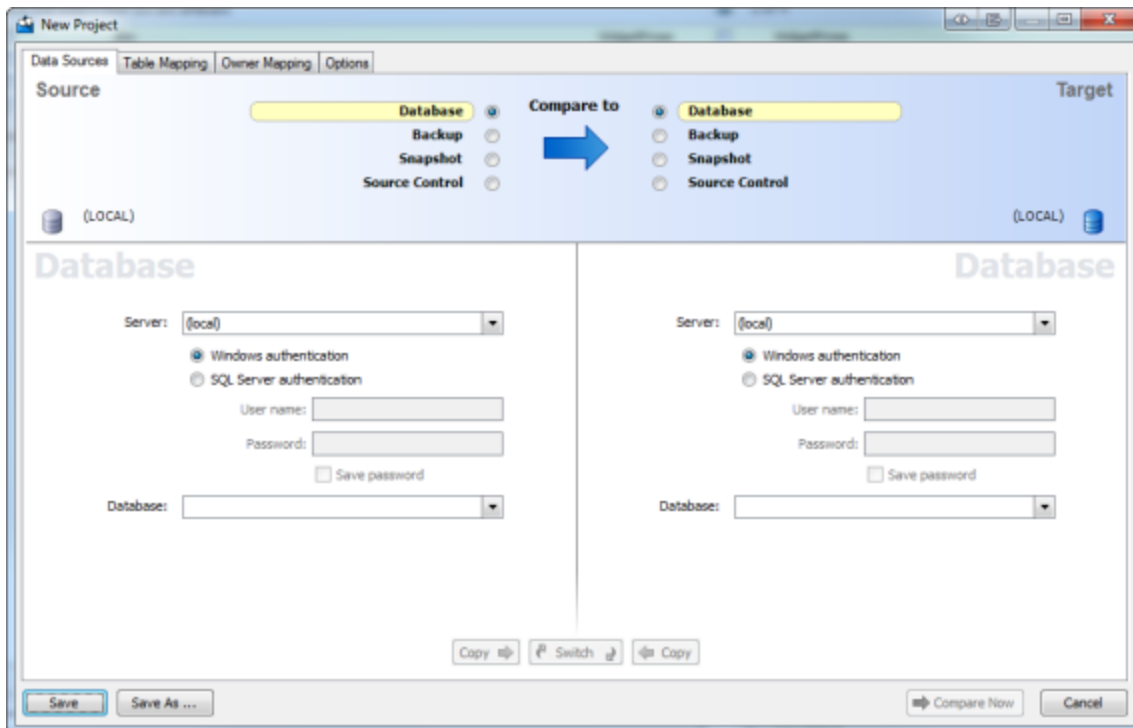


(New Project).

If an unsaved project is currently open, you will be prompted to save when you click



The Project Configuration dialog box is displayed:

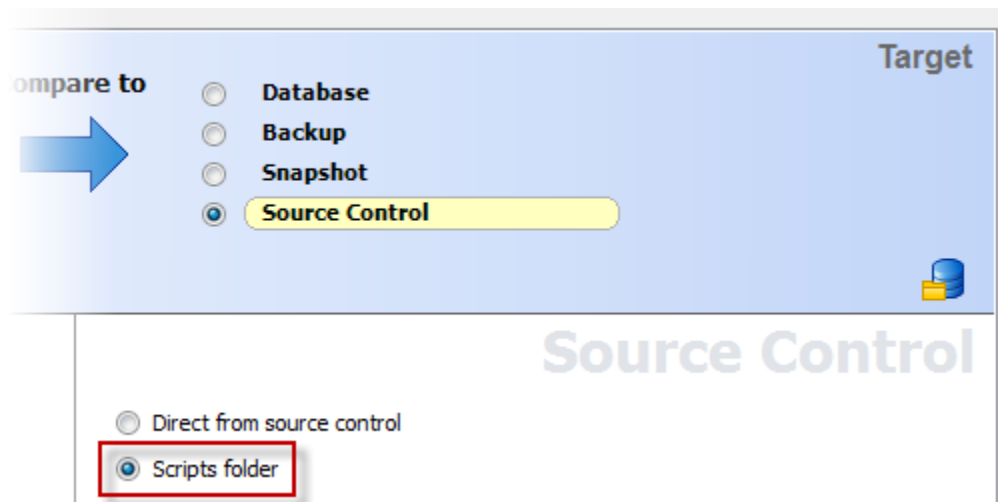


2. Under Server, select the server you're using.
3. Under Source, select *Database*, and in the **Database** box type or select *SprocketDevelopment*.

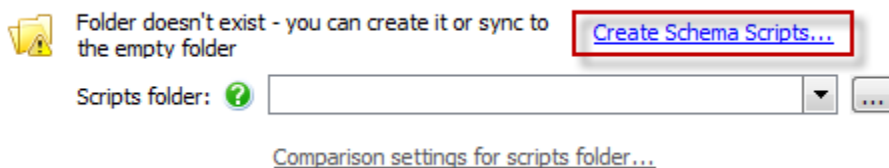
SprocketDevelopment is the new version of the database, following a development cycle.

You must now create a scripts folder representing the schema of the production database.

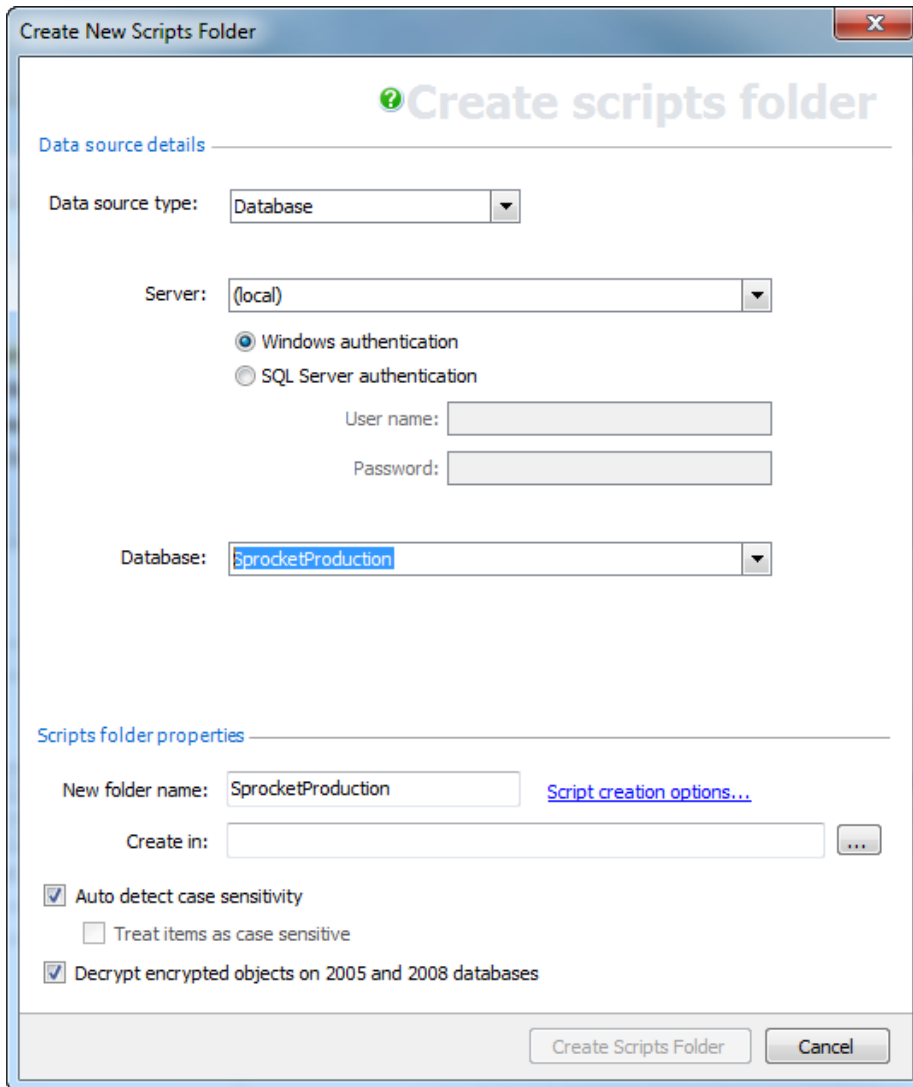
4. Under Target, select *Source Control*, then select **Scripts folder**:




5. Click **Create schema scripts**:



The **Create New Scripts Folder** dialog box is displayed:

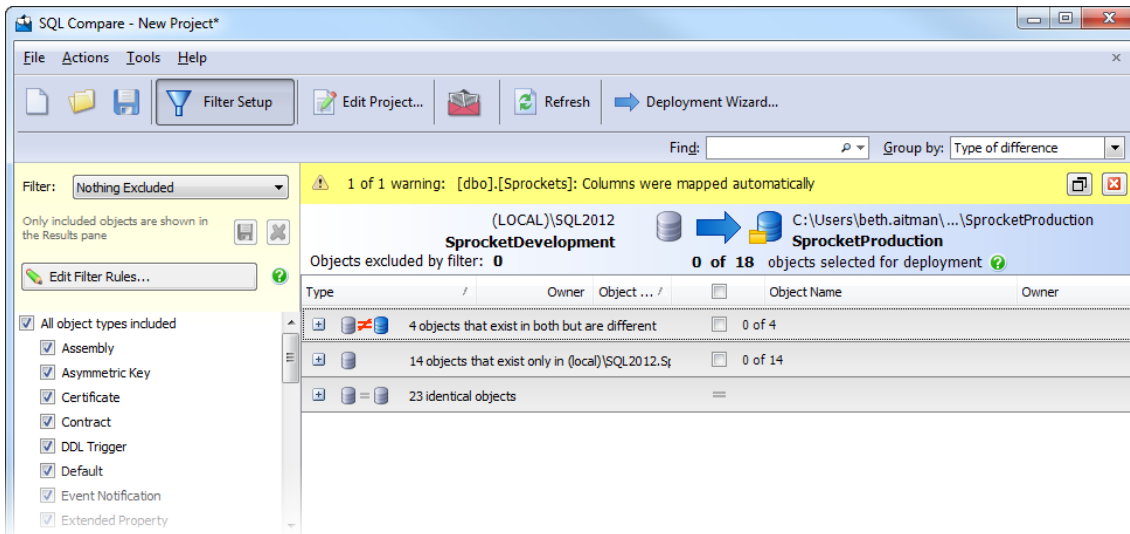


6. Ensure that the **Data source type** is set to *Database*.
7. Type or select *SprocketProduction* in **Database**.  
If the database is not displayed in the **Database** list, right-click in the **Database** box and click **Refresh**, or scroll to the top of the list and click **Refresh**.
8. Click in the **New folder name** box. SQL Compare automatically supplies a name for the scripts folder.  
The default value is the name of the data source.  
In this example, use the default name *SprocketProduction*.  
In **Create in**, type the path for the folder you want to create, or click  **Browse** to browse to its location.
9. If you are using a source control system, this may be the folder designated as your working folder.
10. Click **Create Scripts Folder**.  
The schema is saved in the specified folder as a set of object creation script files.  
The folder you created is now shown as the target of the comparison on the Project Configuration dialog box.
11. Click **Compare Now**.

A message dialog box is displayed. If you selected the **Close dialog box on completion** check box last time you ran a comparison, SQL Compare closes this message dialog box automatically.

### 3. Select objects to deploy

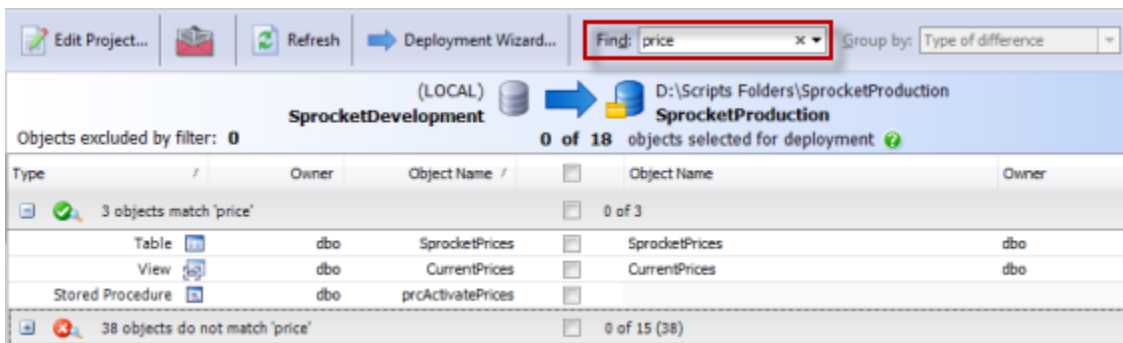
The comparison results are displayed in the main window:



To update the production database, you must create a deployment script. Verify that all the objects you want to modify are present in the comparison results, and select them for deployment.

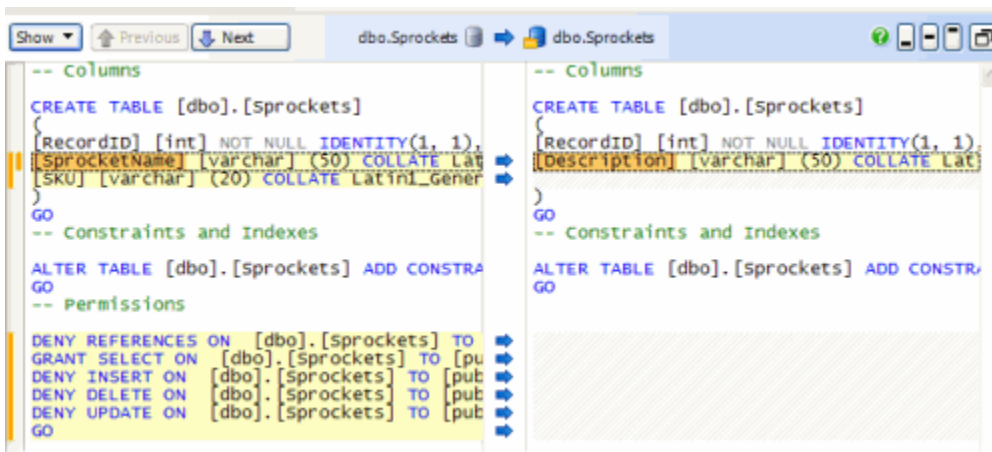
To search for objects, type the search text in the **Find** box. SQL Compare searches object names and owner names.

In this example, search for all objects that contain "price" by typing *price* in the **Find** box.

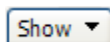


The objects are now grouped by whether they match or do not match the find text. To clear the **Find** box, click the **X** button; all the objects are displayed.

You can view a side-by-side, color-coded listing of the differences in the object creation scripts, by clicking an object. For example, if you click the *Sprockets* table, you can see the differences for this table:



The two versions of the line are shown one on top of the other in the Line Differences bar. This is especially useful when the lines are too long to view all the text; you can see more of each line in the Line Difference bar. If the Line Differences bar is not displayed, click



and select **Line Differences**

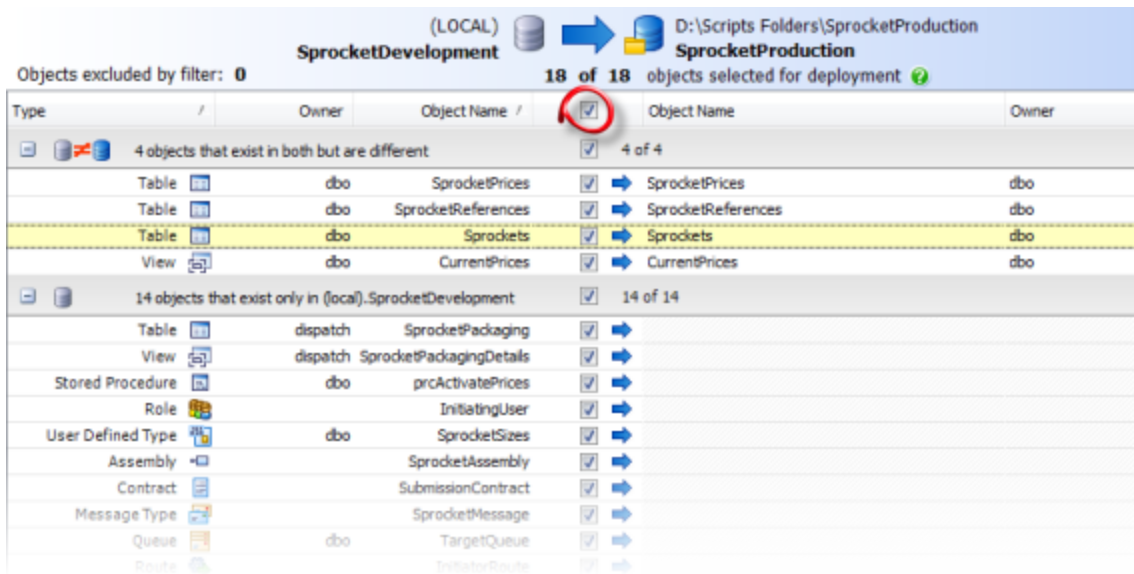


For full details of how to use the comparison results window, see:

- Viewing the comparison results
- Viewing the SQL differences

In this example we will deploy all objects.

In the central column of the upper (Results) pane, select the top level check box to include all objects in the deployment:



Type	Owner	Object Name	Object Name	Owner
4 objects that exist in both but are different				
Table	dbo	SprocketPrices	SprocketPrices	dbo
Table	dbo	SprocketReferences	SprocketReferences	dbo
Table	dbo	Sprockets	Sprockets	dbo
View	dbo	CurrentPrices	CurrentPrices	dbo
14 objects that exist only in (local).SprocketDevelopment				
Table	dispatch	SprocketPackaging		
View	dispatch	SprocketPackagingDetails		
Stored Procedure	dbo	prcActivatePrices		
Role		InitiatingUser		
User Defined Type	dbo	SprocketSizes		
Assembly		SprocketAssembly		
Contract		SubmissionContract		
Message Type		SprocketMessage		
Queue	dbo	TargetQueue		
Route		InitiatorRoute		

#### 4. Create a deployment script

When you have selected the objects to deploy, click



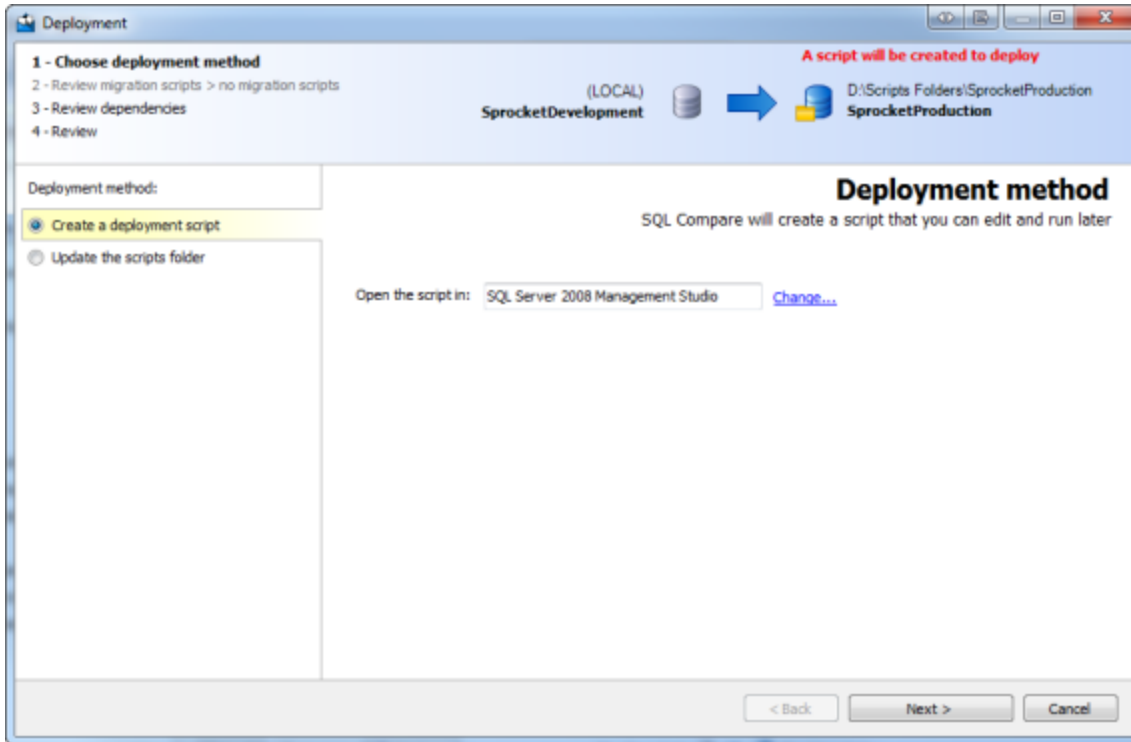
**Deployment Wizard.**

##### 1. Choose deployment method

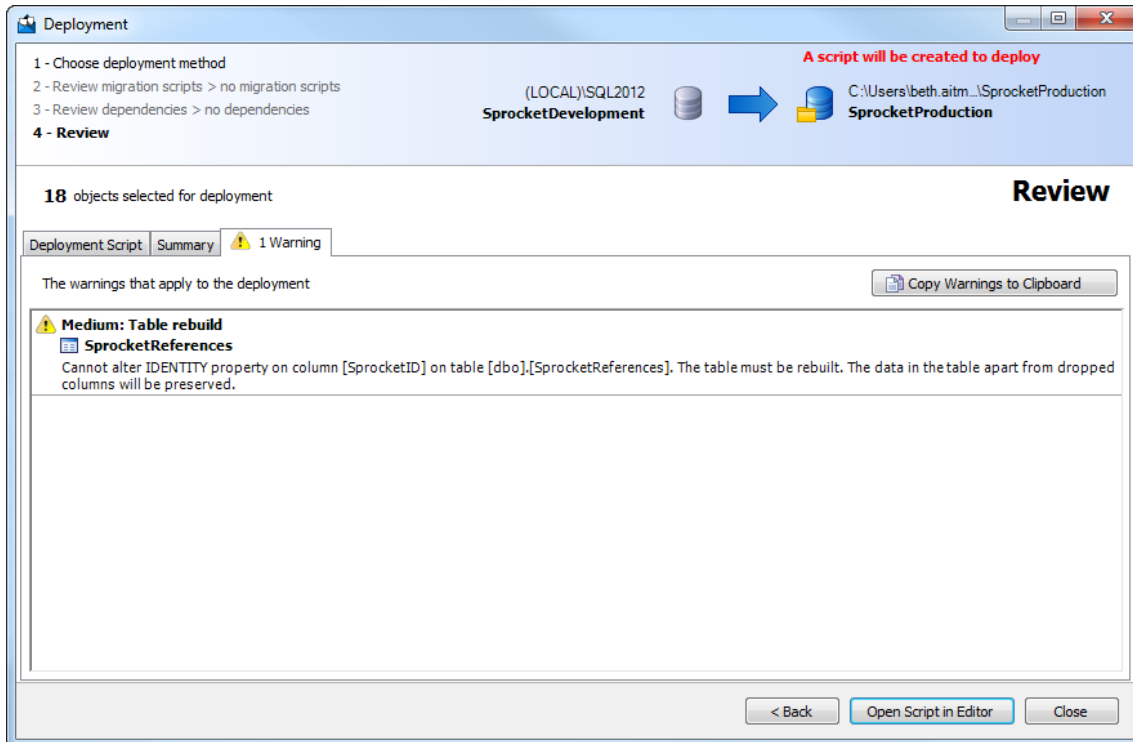
When a scripts folder is the target, you can:

- Create a deployment script to update the database from which the scripts folder was created
- Modify files in the scripts folder directly

In this example, we will create a deployment script. The script can then be run on the production database to update it with the development changes.



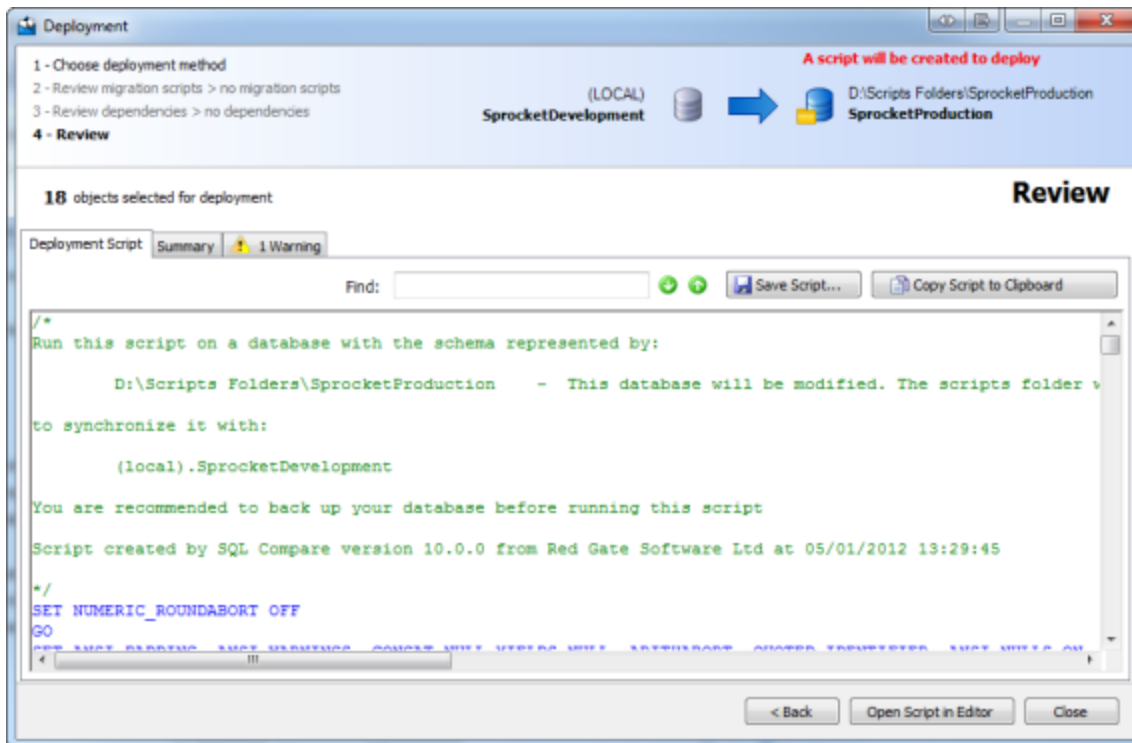
1. Ensure that **Create a deployment script** is selected.
  2. Click **Next**.
- The Warnings tab is shown:



## 2. Review script

When you have reviewed the script, you can open it in your SQL editor, or save a copy of the script.

1. Click **Open Script in Editor**.  
The script is opened in your default SQL editor, and can be run on the *SprocketProduction* database to update it with the development changes:



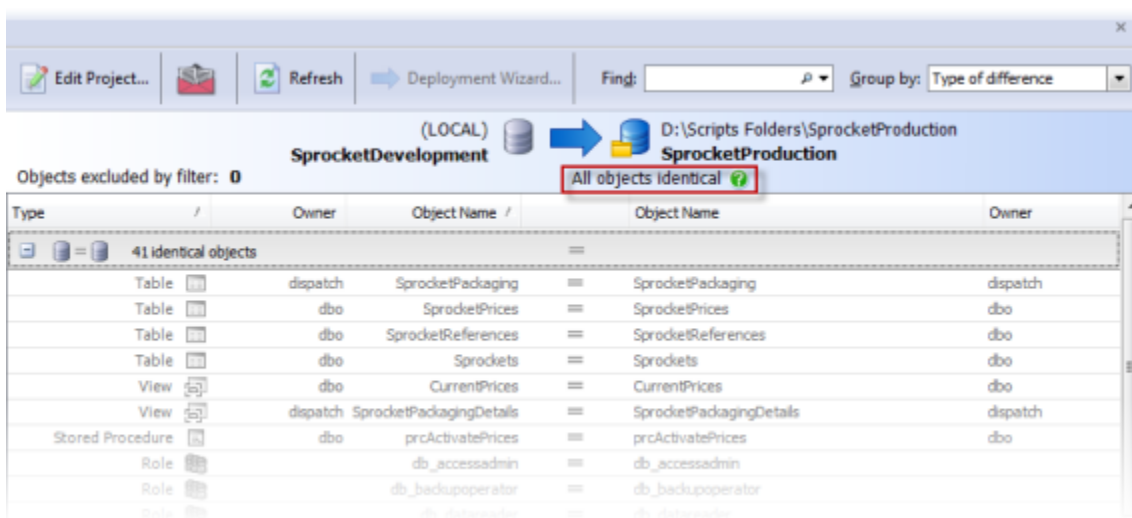
2. Run the script.

To verify that the deployment was successful, set up a new project comparing *SprocketDevelopment* and *SprocketProduction*. If you have not saved the project, you will be prompted to do so when you click



It is not necessary to save the current project.

As the databases are now identical, *All objects identical* is shown in the direction bar:



# Troubleshooting

## Common issues

- SQL Compare shows identical objects as different
- Comparison seems slow
- Errors in scripts folders
- Can't compare encrypted text in SQL Server 2005 or 2008
- Deploying a SQL 2000 compatible database from a SQL 2005 database using SQL Compare
- When does the deployment process rebuild tables?

## Error messages

- A duplicate object name has been found
- HTML reports are generated for identical comparisons
- Could not enlist in a distributed transaction
- Could not start a transaction for OLE DB provider
- Error Comparing DB1 vs DB2. Cannot insert the value NULL into column 'YourColumn', table 'database.dbo.tmp\_rg\_xx\_MyTable'; column does not allow nulls. INSERT fails
- SQL Server doesn't exist or access is denied
- The DEFAULT\_SCHEMA clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys
- The operation could not be performed because the OLE DB provider 'SQLOLEDB' was unable to begin a distributed transaction...
- This SQL Server has been optimized for X concurrent queries. This limit has been exceeded by X queries and performance may be adversely affected
- User or role already exists in the current database / The login already has an account under a different user name

## Common issues

- SQL Compare shows identical objects as different
- Comparison seems slow
- Errors in scripts folders
- Can't compare encrypted text in SQL Server 2005 or 2008
- Deploying a SQL 2000 compatible database from a SQL 2005 database using SQL Compare
- When does the deployment process rebuild tables?

## SQL Compare shows identical objects as different

The results pane that shows the differences between objects that exist in both databases is showing objects that do not have any differences.

For example, select one of the views and see (view  $\neq$  view) at the top but the details do not show any lines of the view that are different.

The issue is a textual vs semantic comparison issue.

If the object is listed in the 'identical' group in the top grid, but when you click on the object it is shown with differences in the bottom grid, then the difference is probably related to the two types of comparison SQL Compare performs.

When SQL Compare compares a schema, a semantic comparison will be performed (top grid) which will group the objects based on the rules you have applied to the deployment and other basic functionally similar syntax that will automatically be ignored. It is these differences that will be deployed.

The other comparison is a straight textual comparison, that will be displayed in the bottom SQL Differences pane. All textual differences will be highlighted here. We can try match the rows as best we can, but each method will have its limitations and we don't think it would be right to reorder the SQL at this stage.

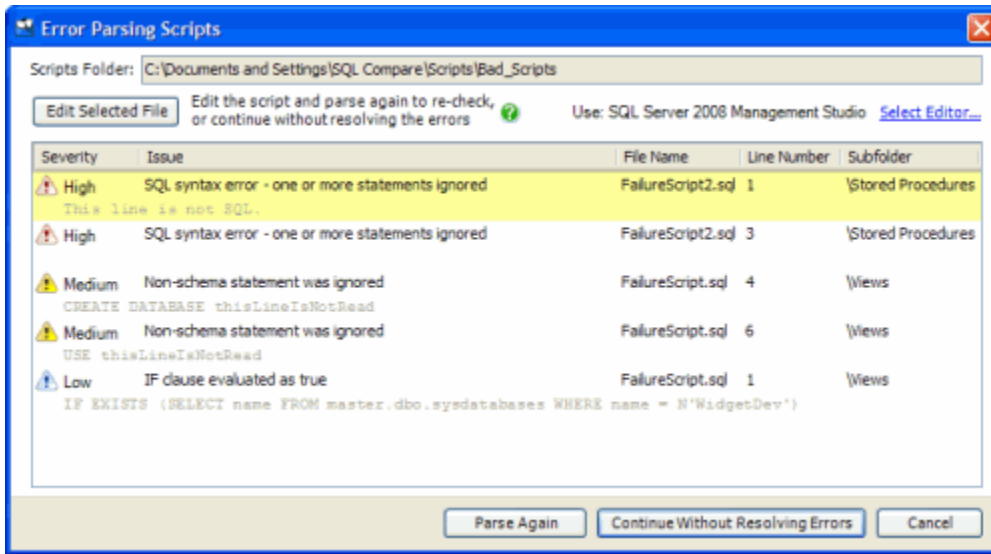
## **Comparison seems slow**

The process of comparing SQL Server 2005 and 2008 databases may seem slow, especially compared to previous SQL Compare.

SQL Compare 7.1 reintroduced the ability to decrypt encrypted objects on SQL Server 2005 and 2008 databases. If the database contains a large number of objects that reside in SQL Server's SYSCOMMENTS table, such as stored procedures and user-defined functions, SQL Compare has to read through the code for all of these objects whether they are encrypted or not. The performance will improve after turning off the "Decrypt encrypted objects on 2005 and 2008 databases" option in the project options tab.

## Errors in scripts folders

If SQL Compare encounters an error comparing a scripts folder, the **Errors Parsing Scripts** dialog box is displayed:



This is most likely to occur if the scripts folder you are using has been edited or was not created by SQL Compare.

Where possible, the line with the error is shown, along with the location of the script file. The following errors are listed:

- SQL syntax error  
SQL Compare could not successfully parse the object creation script. Affected objects may not appear in the comparison results.
- Non-schema statement ignored  
A statement that does not affect the schema was found and ignored.
- Reference not found  
An ALTER TABLE CONSTRAINT statement refers to a table or constraint that cannot be found.
- IF clause evaluated as true  
SQL Compare considers only the first block of code in IF statements. Schema changes made in ELSE conditions do not appear in the comparison results.

Proceeding without resolving these errors can result in comparison results that do not accurately reflect the structure of the database. A deployment script generated for these data sources could fail or have unexpected results.

If a deployment fails, in most circumstances changes are rolled back. SQL Compare uses *transactions* to do this. However, there are some circumstances in which this is not possible.



## Can't compare encrypted text in SQL Server 2005 or 2008

Why can't I successfully compare encrypted object text when comparing SQL Server 2005 and 2008 databases?

SQL Compare 3.0 introduced seamless decryption of encrypted stored procedures, views, and user-defined functions in SQL 2000, however, this functionality did not work with SQL Server 2005 because of improved encryption methods in the SQL Server engine.

If you have got a SQL Server 2005 database and you are given the message "the object's text is encrypted" in the place of a SQL query in the SQL code window, upgrading to SQL Compare 7.1 and up will solve this issue, provided the option to compare encrypted objects is selected in the project options.

Like in SQL 2000 object decryption, elevated permissions on the database server are required to decrypt database objects. This typically means that the account you use to connect to the database with will have to be a member of the SYSADMIN or SECURITYADMIN roles.

To test whether or not you have high enough permissions on SQL Server 2005 and 2008, running these queries successfully would be a good indication:

```
DBCC DBINFO  
DBCC PAGE
```

## Deploying a SQL 2000 compatible database from a SQL 2005 database using SQL Compare

It's possible to convert objects from or migrate a SQL 2005 database to a SQL 2000 database using SQL Compare.

SQL Compare creates a deployment script in a way that is compatible with the database being deployed. For example, if the database resides on the SQL 2000 platform, the database scripts will be created in the SQL 2000 syntax.

There is a workaround to allow a SQL 2005 database to be deployed with a SQL 2000 server. In order to create a deployment script that will run on SQL Server 2000, first you will need a SQL Server 2000. Creating a new database on a SQL Server 2005 and setting the compatibility level to 80 will still result in a SQL Server 2005 script.

First, create a new database on the SQL Server 2000 instance using Enterprise Manager, SSMS or writing a CREATE DATABASE [databasename] query.

Launch SQL Compare and choose your SQL 2005 database as the source and specify the empty SQL 2000 database as the target.

SQL Compare will produce a script to CREATE all objects in dependency order, and use syntax that is compatible with SQL Server 2000. 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 can't be created, the stored procedure can't be successfully scripted. Analyzing your databases for these conditions first is recommended.

In addition to creating a brand-new backwards-compatible database using this method, you may also create SQL Server 2000-compatible migration scripts for an existing SQL 2000 database using the same method.

## When does the deployment process rebuild tables?

SQL Compare will alter existing tables whenever possible when it deploys database schema, but sometimes the entire table must be dropped and recreated. This involves creating a new table, copying all of the data to the new table, and dropping the old table.

SQL Compare will do this in these circumstances:

- Changing an IDENTITY seed or increment value, or dropping the IDENTITY property from a column
- Adding a column in the middle of a table when the 'force column order' option is enabled
- When you add, alter or drop PERSISTED computed columns
- When it's impossible to implicitly cast one data type to another (e.g. DECIMAL to NUMERIC)
- Changing the filegroup specification for a table
- Changing partitioned columns
- Adding a column that has no default and does not accept NULL values
- Azure: when an ALTER COLUMN will cause a Primary Key or clustered index to be dropped and data exists in the table

## Error messages

- A duplicate object name has been found
- HTML reports are generated for identical comparisons
- Could not enlist in a distributed transaction
- Could not start a transaction for OLE DB provider
- Error Comparing DB1 vs DB2. Cannot insert the value NULL into column 'YourColumn', table 'database.dbo.tmp\_rg\_xx\_MyTable'; column does not allow nulls. INSERT fails
- SQL Server doesn't exist or access is denied
- The DEFAULT\_SCHEMA clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys
- The operation could not be performed because the OLE DB provider 'SQLOLEDB' was unable to begin a distributed transaction...
- This SQL Server has been optimized for X concurrent queries. This limit has been exceeded by X queries and performance may be adversely affected
- User or role already exists in the current database / The login already has an account under a different user name

## A duplicate object name has been found

SQL Compare displays this message when you compare a database on a SQL Server that uses case-sensitive sort order, and you have not selected the **Treat items as case sensitive** project option.

SQL Compare also displays this message if you create a snapshot of a database on a SQL Server that uses case-sensitive sort order, and you have not selected the **Treat items as case sensitive** check box on the **Create Database Snapshot** dialog box.

## HTML reports are generated for identical comparisons

If you are using the SQL Compare command line interface to automate schema comparison and report generation, you may have noticed that a report is still generated when the schemas are identical.

Currently it is not possible to set the job to only create the report if the schemas are different, as the report will be generated before the error code indicating no differences is returned. This results in a somewhat unhelpful blank report being created in the folder.

As a possible workaround, you could configure the batch file to delete the report if the schemas are identified as identical. .

Errorlevel 63 will be returned if the schemas are the same, which can be used to delete the report. Here is an example batch file:

```
"C:\program files\red gate\sql compare 7\"sqlcompare /s1:(local) /s2:(local)
/db1:TempA /db2:TempB /report:"E:\Temp\SQL Compare Reports\report.html" /rt:simple
/force

IF %ERRORLEVEL% == 0 GOTO end

IF %ERRORLEVEL% == 63 GOTO identical

:identical

del "E:\temp\SQL Compare Reports\report.html"

:end
```

## Could not enlist in a distributed transaction

SQL Compare may display this message if you're updating a stored procedure that references objects across a linked server, and the database system on the linked server does not support the same transaction isolation levels as SQL Server.

To prevent other schema modifications from taking place at the same time as the deployment, SQL Compare sets the transaction isolation level to SERIALIZABLE. To run the deployment SQL script on the target database, you may need to change the transaction isolation level. To do this, open the script in your SQL application, and find this line:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

Change this to:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```

You should take precautions to make sure no changes are made to the target database structure when you run the deployment SQL script.

## **Could not start a transaction for OLE DB provider**

SQL Compare displays this message when your source database contains an object that references a linked server, but the linked server is not defined on the target server.



**Error Comparing DB1 vs DB2. Cannot insert the value NULL into column 'YourColumn', table 'database.dbo.tmp\_rg\_xx\_MyTable'; column does not allow nulls. INSERT fails**

This error occurs when adding a new column to an already populated table, and the new column does not allow null values.

To fix this:

- assign a default value to the column in the source database, or
- set the column to allow nulls, and then set it back after you have the correct data inserted.

## SQL Server doesn't exist or access is denied

SQL Compare can't connect to the SQL Server. Try the following to rectify the problem:

1. Verify 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.

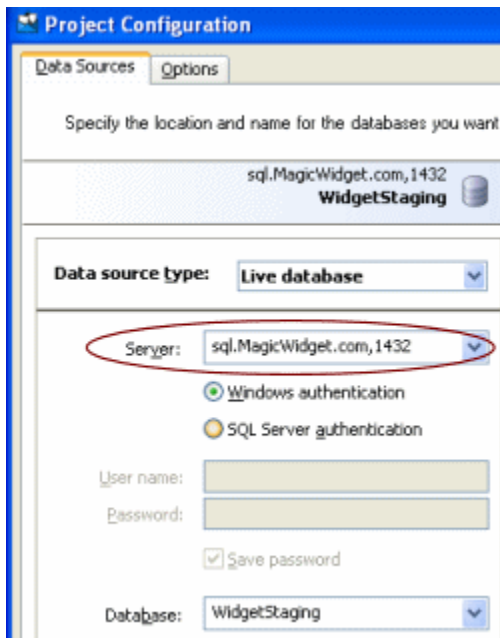
If the SQL Server is online, verify that you are connecting to the correct port.

If your SQL Server is not running on the default port (1433), type the following in **Server** on the **Project Configuration** dialog box:

<ServerName>,<Port>

where ServerName is the name of your SQL Server and Port is the number of the port on which your SQL Server is running.

For example:



If you are sure that you are connecting to the correct port, force SQL Compare to use the TCP network protocol when it makes the connection, by typing the following in **Server** on the **Project Configuration** dialog box:

TCP:<ServerName> For example:

**Project Configuration**

Data Sources Options

Specify the location and name for the databases you want

TCP:sql.MagicWidget.com  
**WidgetStaging**

Data source type: Live database

Server: TCP:sql.MagicWidget.com

Windows authentication  
 SQL Server authentication

User name:   
Password:

Save password

Database: WidgetStaging

## **The DEFAULT\_SCHEMA clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys**

This error occurs when deploying, and causes the deployment to fail.

The DEFAULT\_SCHEMA clause cannot be used with a Windows group or with principals mapped to certificates or asymmetric keys.

This is because 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.

To fix this, note the name of the group in the error dialog, and remove that SQL Server user from the deployment.

## The operation could not be performed because the OLE DB provider 'SQLOLEDB' was unable to begin a distributed transaction...

### Error

When deploying to a database, either through SQL Compare, SQL Packager or when running a SQL script produced by them, the following error message may be displayed:

The operation could not be performed because the OLE DB provider 'SQLOLEDB' was unable to begin a distributed transaction.  
[OLE/DB provider returned message: New transaction cannot enlist in the specified transaction coordinator. ]  
OLE DB error trace [OLE/DB Provider 'SQLOLEDB' ITransactionJoin::JoinTransaction returned 0x8004d00a].

This error message is displayed when a SQL Server programmability object such as a stored procedure or function references a linked server or distributed query connecting to another server. The script fails because the transaction isolation level selected to run the SQL Compare script is incompatible with distributed queries. SQL Compare uses TRANSACTION ISOLATION LEVEL SERIALIZABLE to protect the schema from interference while the update is performed. For this reason, we will not be changing the isolation level to be compatible with distributed queries.

### Workaround

#### 1. Editing a migration script

- Save a migration script to a file using the deployment wizard of SQL Compare after a database comparison by clicking the **Save SQL Script** button.
- Once the script is saved to disk, locate the line: `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` and change this to `SET TRANSACTION ISOLATION LEVEL READ COMMITTED`. This will allow distributed query DDL to be submitted.
- Run the script in SQL Server Query Analyzer or Management Studio.

This can cause problems by allowing competing DDL operations to run at the same time as the SQL Compare script.

#### 2. Disabling transactions in deployment scripts

- Go into the project settings
- Click the options tab
- select 'do not use transactions in deployment scripts'.

This eliminates the issue by not breaking the script up into transactional units.

You will lose the rollback functionality if an error occurs during the running of the script, possibly leaving the database in an 'unknown' deployment state.

**This SQL Server has been optimized for X concurrent queries. This limit has been exceeded by X queries and performance may be adversely affected**

When comparing and/or deploying, the following error may be returned by SQL Server:

"This SQL Server has been optimized for x concurrent queries. This limit has been exceeded by x queries and performance may be adversely affected."

This may make it seem as if SQL Compare or Data Compare is making many connections to the database, however, SQL Data Compare and SQL Compare will only make a single connection to each database it is comparing.

If you want to check to see exactly what connections are being made to SQL server, by Red Gate tools or other processes, you can check the current connections by running `sp_who` (see books online) in a query editor in the context of the master database. Using that, you should be able to see which connections are currently open and what is using them.

## **User or role already exists in the current database / The login already has an account under a different user name**

If a user needs to be added to a database as part of the deployment, then the deployment will fail if the login has already been granted access to the database under a different username. The error returned by SQL Compare is either "User or role already exists in the current database" or "The login already has an account under a different user name".

This has been identified as an issue because SQL Compare should really revoke database access to the old user name before granting access to the same login under a new username. The suggested workaround is to select the users and roles that exist only in the database being deployed (database 2) and deploy only these objects, which has the effect of revoking the login access to the database first. After this deployment, a second deployment of the remaining objects should succeed and the login will once again be granted access to the database under the new username.

## Release notes and other versions

<b>Version 11.4 (current)</b>	December 21st, 2015	<a href="#">Release notes</a>	<a href="#">Documentation</a>
<b>Version 11.3</b>	August 24th, 2015	<a href="#">Release notes</a>	
<b>Version 11.2</b>	May 7th, 2015	<a href="#">Release notes</a>	
<b>Version 11.1</b>	January 29th, 2015	<a href="#">Release notes</a>	
<b>Version 11.0</b>	October 6th, 2014	<a href="#">Release notes</a>	
<b>Version 10.7</b>	April 23rd, 2014	<a href="#">Release notes</a>	<a href="#">Documentation</a>
<b>Version 10.4</b>	June 3rd, 2013	<a href="#">Release notes</a>	
<b>Version 10.3</b>	March 11th, 2013	<a href="#">Release notes</a>	
<b>Version 10.2</b>	May 16th, 2012	<a href="#">Release notes</a>	
<b>Version 10.0</b>	December 15th, 2011	<a href="#">Release notes</a>	
<b>Version 9.0</b>	February 17th, 2011	<a href="#">Release notes</a>	<a href="#">Documentation</a> <a href="#">PDF</a>
<b>Version 8</b>	October 28th, 2010	<a href="#">Release notes</a>	<a href="#">Documentation</a> <a href="#">PDF</a>
<b>Version 7</b>	July - Sep 2008	<a href="#">Release notes</a>	<a href="#">Documentation (PDF)</a>
<b>Version 6</b>	Aug - Oct 2007	<a href="#">Release notes</a>	<a href="#">Documentation (CHM)</a>

If you need to install an old version of SQL Compare, go to [Download old versions of products](#).



# SQL Compare 10.7 release notes

April 23rd, 2014 [Download](#)

## New features

- Supports connecting to SQL Server 2014 databases
- Improvements to the options API for Comparison SDK. This includes breaking changes which will require you to update your code to support the new SDK version – see [Breaking changes in SQL Comparison SDK 10.5](#) for further details.

## Fixes

- SC-6830: SQL Compare no longer raises an error when the THROW syntax does not have an argument
- SC-6032: Ignore system named constraints now correctly ignores default system named constraints
- SC-5850: SQL Compare no longer causes a Utils.EncloseInSquareBrackets.NullReferenceException error
- SC-6885: SQL Compare no longer throws a InvalidCastException when attempting to register a scripts folder containing filetables
- SC-6770: Fixed an issue where the Deployment wizard could skip the 'Reviewing dependencies' step
- SC-6664: SQL Compare no longer causes an error on the POISON\_MESSAGE\_HANDLING keyword in queues
- SC-6613: SQL Compare no longer causes an error on the 'MERGE' keyword when target is table object
- SC-6942: SQL Compare no longer demands .NET 3.5 on Windows 8 machines
- SC-2907: Fixed an issue where SQL Compare might not update a column's data type if the column is a foreign key
- SC-6860: The SQL Source Control history dialog no longer displays rogue string – this fix requires a future version of SQL Source Control (v3.5.7.450 and above)
- SC-6669: Fixed an issue which was blocking registration of a database when the user has limited rights
- SC-6147: Clustered index changes are now handled correctly when working with a SQL Azure database as a target.
- SC-2656: Users created from windows logins are now scripted properly
- SC-6564: Script no longer GRANTS permissions to the wrong columns after a column is dropped
- SC-6894: Syntax highlighting has been updated for ROWS, FIRST and ONLY keywords
- When selecting a source control system as the data source the revision number is no longer changed to 'Head' when a different source control repository is selected
- SC-6317, SC-6402 - Drop and Create instead of Alter option now works correctly with SOC data sources
- SC-6381 - NullReferenceException caused by rare dependency situations fixed
- SC-5365 - NullReferenceException caused by some queue differences fixed
- SC-6575, SC-6291 - Ignoring system-named constraints no longer causes us to drop system-named FKs on table rebuild
- SC-2907 - Foreign key migrations where only the target table is selected no longer fail when target data type has changed
- SC-6439 - temporary inline indexes during table rebuilds now no longer have a name collision with the final inline index
- SC-6493 - trigger dependency on new column of existing table now respected
- SC-5951 - LOCK\_ESCALATION setting now compared and deployed
- SC-5615 - some causes of this exception involving duplication of triggers in script folders have been fixed
- SC-6781 - overflow on reading extremely large identity values from script folders fixed
- SC-6468, SC-6471, SC-6475, SC-6316, SC-5607, SC-6394, SC-6401, SC-5379, SC-5718, SC-5812, SC-5849, SC-5914, SC-5916, SC-6137, SC-6153, SC-5527 - progress dialog made more robust against a range of conditions
- SC-6301, SC-6764, SC-6234, SC-5342, SC-5974, SC-5770, SC-5977, SC-5966, SC-6230, SC-6093 - documentation inconsistencies and omissions fixed, particularly around worked examples and filter syntax
- SC-4640 - handling of route lifetimes improved
- SC-6566 - TFS connection issue when using migration scripts fixed
- SC-6562 - A situation where encrypted objects weren't decrypted is now fixed
- SC-6055 - datetimeoffset(0) now consistently displayed when read from script folders
- SOC-3873 - money type no longer overflows on very large numbers when reading from script folders
- SC-6056 - fixed occasional exception in picking which migration scripts to enable by default
- SC-6073 - script folder deployment issue when changing from inline to explicit check constraint fixed
- SC-6072 - fixed failure to load projects with custom script folder layout options
- SC-6085 - fixed a crash on getting latest changes with SOC with the sysname datatype in a table
- SC-5599 - rare Cryptographic Exception when loading some v8 or earlier projects fixed
- SC-6099 / SC-5184 - fixed a problem with user deployment with mapped schemas
- SC-6019 - fixed a crash in the deployment wizard when script generation fails
- SC-5215 - fixed a crash where case sensitivity got unset in the middle of a snapshot comparison
- SC-6066 - /include:Identical command line option now no longer incorrectly applies filter rules
- SC-6112 - warning to replace deprecated XML file from SOC now displays file name correctly
- SC-5677 - /assertidentical and /report command line options can now be used together
- Recently used repositories now show their migrations folder (if they have one, in a tooltip (to make it clear what the difference is between repositories with the same main path but different migration folders)
- Priority of object selections from filters vs individual selections when using project on the command line should now be identical to UI behaviour (previously some filters took priority incorrectly)
- Foreign key dependencies on reference primary keys now enforced for script folders
- Including static data in a SQL Compare deployment script no longer also includes the SQL Data Compare comment header

## Known issues

- Other new features in SQL Server 2014 (including In-Memory Tables) are not currently supported.
- SC-6986: A "Data Compression enum value not recognised" error can occur when viewing a table with a clustered columnstore index

SQL Compare 10.5 and 10.6 were internal and beta builds that were not publicly released.

# SQL Compare 10.4 release notes

June 3rd, 2013

## New features

- This release of SQL Compare contains significant performance improvements - up to 50% faster on script folder registration, 60% faster on the registration and 70% faster of the comparison of large databases.
- There is a new option to ignore WITH ENCRYPTION in textual objects (views, functions, stored procedures etc)

## Fixes

- Improved UI stability (fixed several race conditions that occasionally produced null reference exceptions, index out of range conditions or invalid operation exceptions in the UI)
- SC-5862: Save project menu item now reliably disabled when no project open
- SC-6215: stored procedure dependency now working better from command line
- SC-6165: identical objects no longer included without specifying /include:Identical when the /report switch is used
- SC-6216: command line no longer generates .NET crypto warning in event viewer
- SC-5347: database extended properties now have an icon and title on interactive HTML reports
- SC-5766: dropping objects which are in full text indexes will now drop references to them more reliably from script folders
- SC-6071: CLR types and partition scheme unbind now interacts correctly with default constraints
- SC-6147: now ensures more tables retain clustered indexes during table rebuilds on Azure
- SC-5693: function dependency which was ignored when drop and create option on now picked up correctly
- SC-6250: dependency ordering issue between stored procs and view triggers fixed
- SC-6171: guest user no longer capitalised in revoke connect statements on case sensitive servers
- SC-5850: nullref when deploying some changes to script folders now fixed
- SC-6020: fixed a problem with the drop-and-create-instead-of-alter option applied to script -> script deployments
- SC-6275: drift detection will now do a case-sensitive comparison on case-sensitive servers
- SC-4754: assembly dependencies based on assembly content now displayed with other object dependencies
- SC-6165: /report switch no longer forces identical objects to be included even when /exclude:identical is specified
- SC-5384: explains situation rather than throwing an exception when clipboard locked by another application
- SC-5551: now no longer crashes on startup where PROCESSOR\_ARCHITECTURE env variable doesn't exist
- SC-5657: tables with columnstore indexes now compare correctly

## SQL Compare 10.3 release notes

### Visual Studio database project support

This release adds initial support for Visual Studio database projects, as either the source or the target of comparisons and deployments. To do this, use Source Control -> Scripts Folder and navigate to the folder containing the .sqlproj file.

### Stability improvements

SQL Compare 10.3 also includes fixes for some issues:

- Statistics with multiple columns now no longer cause problems in script folders
- Permissions for search property lists and sequences are now correctly written to scripts folders
- SOC-4253 / SC-6194 - parsing of OFFSET clause in script folders improved
- SDC-1540 - parsing of varchar(max) columns from script folders with a primary key of type 'real' fixed
- Parsing of CONTAINSTABLE with extra parens now fixed

### Project compatibility changes

Projects saved by SQL Compare 10.3 aren't readable by SQL Compare 10.2 and earlier, due to a change in the file format.

# SQL Compare 10.2 release notes

## SQL Compare 10.2.3

### New feature: scripts folder static data support

If you're currently running both SQL Compare and SQL Data Compare to generate a SQL deployment script from a scripts folder, you will no longer need to run both: SQL Compare, with its new command line option `/include:StaticData` will now include static data (which you will still have to deploy to the script folder with SQL Data Compare or commit with SQL Source Control) in the deployment script that it generates/runs. It even puts the static data changes into the same transaction as the schema changes.

### Known issues:

Some users who have customized the folder layout of their script folders (i.e. which folders contain which object types) may encounter a `NullReferenceException` in `ScriptDatabaseInformation.ReadFromXml`.

If you encounter this exception, deleting the file found at `C:\Documents and Settings\<your username>\Local Settings\Application Data\Red Gate\SQL Compare\WriteToFileOptions.xml` should fix the issue – SQL Compare will write out a new `WriteToFileOptions.xml` in the correct format.

You may need to re-enter your custom folder layout in the usual place and save your default settings for this feature again to restore your default settings to your usual layout.

### Improvements:

New command line option `/[include:staticdata]`

Migration scripts from other branches can now be used - just deselect them from the deployment wizard if they are not relevant

Partitioned index warnings now correctly report filegroup names rather than partition scheme names

Allow Page Locks and Allow Row Locks options now supported on view indexes

Command line now correctly handles renamed identity columns in script folder tables

### Bug fixes:

SC-5968: Command line now shows basic help again if no parameters are specified

SC-5598: Individual objects covered by migration scripts are now listed in action list

SC-5657: Columnstore indexes no longer cause an exception during comparison

SC-4824: Parameter defaults with scientific notation and other variant number literals now parse correctly

SC-5743: Names of user defined types in user defined table type definitions are now fully qualified

SC-5815: 'Open in SQL Data Compare' button now works correctly for projects with source-controlled databases

SC-5872: Indexes for temp tables used during rebuild should now correctly prefix names to avoid name collisions

SC-5670: Constraints on tables not in default schema now work correctly with existence check option

SC-5959: Now removes `WITH APPEND` on `CREATE TRIGGER` statements when targeting SQL Server 2012

SC-2020: Default constraints should no longer have different bracketing between SQL Server 2000 and later versions

SC-5898: Typo fix in script creation options

SC-5320: Advice to use migration scripts in warnings will no longer show up when you are already using migration scripts

## SQL Compare 10.2.5

This is a minor release that fixes a few issues found in 10.2.3.1, including the major known issue from the last post regarding custom script folder layouts, but also including:

- recently used repositories now show their migrations folder, if they have one, in a tooltip (to make it clear what the difference is between repositories with the same main path but different migration folders)
- priority of object selections from filters vs individual selections when using project on the command line should now be identical to UI behavior (previously some filters took priority incorrectly)
- foreign key dependencies on reference primary keys now enforced for script folders
- including static data in a SQL Compare deployment script no longer also includes the SQL Data Compare comment header

### Bug fixes:

SC-6072 - fixed failure to start SQL Compare when custom script folder layout options were specified in previous versions

SC-6055 - `datetimeoffset(0)` now consistently displayed when read from script folders

SOC-3873 - money type no longer overflows on very large numbers when reading from script folders

SC-6056 - fixed occasional exception in picking which migration scripts to enable by default

SC-6073 - script folder deployment issue when changing from inline to explicit check constraint fixed

SC-6085 - fixed a crash on getting latest changes with SOC with the sysname datatype in a table

SC-5599 - rare `CryptographicException` when loading some v8 or earlier projects fixed

SC-6099 / SC-5184 - fixed a problem with user deployment with mapped schemas

SC-6019 - fixed a crash in the deployment wizard when script generation fails  
SC-5215 - fixed a crash where case sensitivity got unset in the middle of a snapshot comparison  
SC-6066 - /include:Identical command line option now no longer incorrectly applies filter rules  
SC-6112 - warning to replace deprecated XML file from SOC now displays file name correctly  
SC-5677 - /assertidentical and /report command line options can now be used together

# SQL Compare 10.0 release notes

## Version 10.0 - December 15th, 2011

### New features

- Rerunnable scripts - options to add existence checks and DROP/CREATE
- Recently used servers SQL Compare now remembers the last five connections
- Different tables and columns can be mapped, which means renames will no longer be regarded as DROP/CREATE
- Transaction Isolation Level can be specified as an application option
- New create database functionality from the Project Configuration dialog box, so you can quickly create test databases in SQL Compare
- Filters applied to SQL Source Control databases are now picked up by SQL Compare, and applied in the comparison
- You can now connect to SQL Server 2012 (Denali) servers
- New command line switches

For full details, see: [What's new in SQL Compare 10?](#)

### Fixes

- SC-5244 (OPTION OPTIMIZE not recognized by the parser)
- SC-3996 (Disabled triggers now detected)
- SC-5056 (Database level extended properties behave strangely)
- SC-4930 (Compare always tries to enable change tracking even if already enabled (thus the script fails))
- SC-4938 (/Exclude:Additional ignored when deploying from the scripts folder)
- SC-4958 (Command line include/exclude options not working)
- SC-5089 (Foreign keys not scripted when /scriptfile is used from command line (db -> empty script folder))
- SC-4552 (Support: CL behaves differently to UI: with /argfile switch)
- SC-5014 (Table level compression not handled well when script isn't ours and when it is, there is an 'invisible difference')
- SC-5113 (Comparison reporting identical where SQL differences view highlight a difference (DATA\_COMPRESSION = PAGE))
- SC-4772 (SQL Compare not listing dependencies on Stored Procedure)
- SC-5141 (Create snapshot function unavailable if starting Compare from SSIP)
- SC-4937 (Script error changing partition ranges)
- SC-4031 (Snapshot creation date on command line)
- SC-4974 (Commandline doesn't expose script folder creation options)
- SC-5292 (Make it easier to debug the commandline)
- SC-5310 (XML INDEX and PRIMARY XML INDEX on same column cause drop/recreate)
- SC-5243 (/Makescripts fail on case sensitive servers)
- SC-4706 (File format issue with Generate Comparison Results Report)
- SC-4980 (Partition function can be incorrectly read by script reader)
- SC-4633 (Identically-named partition index works in live database but not script reader)
- SC-5309 (Not creating a function in a deployment script)
- SC-5328 (Change tracking re-enabled in alter statement unnecessarily)
- SC-4487 (Circular dependency issue: Table and Table function)
- SC-5242 (Next button does nothing during deployment)
- SC-4674 (Warning for any dropped column)

# SQL Compare 9.0 release notes

Version 9.0 – February 17, 2011

## New features

- SQL Azure support  
For more information, see [Synchronizing to SQL Azure](#).



# SQL Compare 8 release notes

## Version 8.50 – October 28, 2010

### New features

- A [free add-in](#) for SQL Server Management Studio that enables you to set data sources to compare and synchronize from within SQL Server Management Studio
- Fast deployment and Fast deployment projects: synchronization with one click (Pro feature)
- Support of handling databases under SQL Source Control (using the add-in)

### Changes

- SQL Changeset has been deprecated with the release of SQL Compare 8.50.
- New project dialog option Script folder has been renamed to Source Control, Script folder is a sub option of that

## Version 8.2 – May 10, 2010

SQL Compare 8.2 is a minor release, including a number of bug fixes.

These fixes include:

- Better handling of encrypted objects
- SQL Compare now correctly synchronizes the user permission: *View Change Tracking*
- The command line interface now correctly displays script parser errors
- The command line interface now displays a synchronization script correctly when synchronizing two scripts folders
- The interactive HTML report can now be viewed without an internet connection
- Other minor user interface issues

## Version 8.1 – July 16, 2009

SQL Compare 8.1 is a minor release. It introduces the following features:

- **The ability to share projects with SQL Data Compare**  
To open a SQL Compare project in SQL Data Compare, on the Projects dialog box, select a project, right click, and click Launch in SQL Data Compare.
- **Command line changes**  
There are some changes to the command line syntax
- **Support for static data in scripts folders**  
SQL Data Compare 8.0 introduces the ability to store static data in scripts folders. SQL Compare 8.1 can parse these data scripts. Version 8.0 and earlier cannot parse INSERT statements in scripts folders, and therefore displays error messages when comparing data scripts.

Note that SQL Compare cannot create data scripts in scripts folders.

Known issues with SQL Compare 8.1:

- **Installing in a 64 bit environment**  
There is a known issue that in some cases, when upgrading from version 8.0 to version 8.1 in a 64 bit environment, the installer provides incorrect progress messages. The installer displays the progress messages for a clean install, and does not instruct you to exit version 8.0

If version 8.0 is running when you install version 8.1, you may encounter unexpected results.

You are recommended to uninstall version 8.0 before installing version 8.1 in a 64 bit environment.

- **Project compatibility**  
You can open a SQL Compare 7, 6, or 5 project in version 8.1. Projects from earlier versions are not supported.

You cannot open a SQL Compare 8.1 project in earlier versions, including version 8.0

If you open and save a project in SQL Data Compare 8, it is converted to version 8 and cannot be opened in earlier versions.

You can open a SQL Compare 8.1 (or later) project in SQL Data Compare 8.0 or later. You can open a SQL Data Compare 8.0 (or later) project in SQL Compare 8.1 or later.

## Version 8.0 – February 24, 2009

SQL Compare 8.0 is a major update to SQL Compare, providing fine-grained object filtering, improvements to scripts folder support, and many usability improvements. This release also delivers numerous bug fixes and minor enhancements.

Key changes:

- **Filtering rules:** SQL Compare 8.0 introduces fine-grained filtering of database objects. You can now build highly expressive rules to filter out objects with a specific schema, or objects that have a particular prefix. For example, it is now easy to exclude from the comparison result stored procedures that belong to the "marketing" schema and have the "dev\_" or "tmp\_" prefix.
- **Project selection and project saving:** After a great deal of usability testing and research we have changed the way projects are saved. You now have the ability to explicitly save comparison projects, without having to perform a comparison. The project files are also now easier to locate and share.
- **Error reporting for scripts folders:** In SQL Compare Pro version 6 we introduced the ability to compare and synchronize against SQL creation scripts. Some people had extra SQL statements in their creation scripts that were intentionally not considered by SQL Compare. Examples included drop statements and certain alter statements. SQL Compare now provides detailed information on what statements are ignored, and provides warnings about which branches of conditional statements are being chosen.
- **Handling target scripts as live databases:** Prior versions of SQL Compare Pro could synchronize differences to creation scripts. However, as of this version, SQL Compare can also provide a migration script as well. This feature can be used to quickly generate a change script for the differences between two versions of a database schema when the target schema is stored as a set of creation scripts (for example in source control).
- **Backup before synchronization:** Many customers requested the ability to take a database backup automatically before synchronization. This feature is now part of SQL Compare, and you can choose between a native SQL Server backup, a SQL Compare schema snapshot, or a compressed/encrypted backup with Red Gate SQL Backup if available.
- **Improved the way we display find results** by showing database objects that match as well as objects that do not match a find request.

SQL Compare 8.0 also includes some bug fixes, including:

- Improved dependency resolving, including dependencies between defaults and their referenced functions
- Extended properties of certain data types being inconsistent between the live database and script folders
- Many improvements to schema mappings

## SQL Compare 7 release notes

### SQL Compare 7.1 release notes - September 2008

This is a minor update to SQL Compare. Along with a number of bug fixes there are two principal enhancements:

- Extended support for reading encrypted stored procedures and similar objects to SQL Server 2005 and 2008. (Usual dbo user permissions are required for reading decrypted content)
- Improved schema mappings, making it easier to compare objects between different schemas

### SQL Compare 7.0 release notes - July 2008

This is a major upgrade. New features include:

#### Support for SQL Server 2008:

SQL Compare 7.0 supports SQL Server 2008 based on the currently available release candidate. SQL Server 2008 introduces new data types (table valued types, spatial data types, temporal data types), new and enhanced statements like merge, variable initialization and sparse columns, and many other features, all of which are supported by SQL Compare 7.0.

#### Support for reading backup files directly:

This release of SQL Compare also includes a new Professional Edition feature which allows using database backups directly in comparisons. There is no need to restore the database before a comparison, SQL Compare can read and understand the native SQL Server backup files. This includes backups made by SQL Server 2000, 2005 and 2008 (both compressed and uncompressed backups). SQL Compare 7.0 also supports backup files that are created by Red Gate's SQL Backup tool, including compressed and encrypted backups.

There are also many small enhancements that enable SQL Compare to compare and synchronize databases more efficiently. These include

- Full support for ALTER AUTHORIZATION statements
- Using defaults in table rebuilds for columns that no longer allow null values, but are null
- Ability to deactivate a serial number
- Improved full text index support on views
- More warnings when table rebuilds would require column drops
- New options to ignore the database part of synonyms
- Better matching triggers, constraints, and other dependent objects that help identify differences visually

## **SQL Compare 6 release notes**

### **SQL Compare 6.2 - 25th October 2007**

This is a minor upgrade. New features include:

Source control integration using SQL Changeset. This extension to SQL Compare 6.2 Pro Edition allows people who use source control with SQL Compare 6.2 to automatically get latest versions and check out required SQL script files. This release also includes a few bug fixes.

### **SQL Compare 6.1 - 10th August 2007**

This is a minor update to SQL Compare. It introduces a new option to ignore NOT FOR REPLICATION for check constraints, triggers, identity properties and foreign keys. This release also includes a few bug fixes.

### **SQL Compare 6.0 - 15th May 2007**

The major additions to this versions include:

- Ability to read and synchronize to and from object level SQL scripts
- UI improvements for the project selection
- Search functionality in the UI
- General UI enhancements

\* Work in progress pages \*

## SQL Compare 11 beta release notes

We've now released the fourth beta of SQL Compare 11, which you can [download here](#).

SQL Compare 11 is still in development, and we'd really appreciate your feedback.

### Features

This version includes support for memory-optimized tables. SQL Compare can now:

- perform a deployment to create, drop and rebuild memory-optimized tables on SQL Server 2014 databases
- drop and recreate database level event notifications for CREATE TABLE and DROP TABLE statements when memory-optimized tables are deployed

If your deployment includes memory-optimized tables, you need to:

- make sure that the target database has a memory-optimized filegroup
- turn on the **Do not use transactions in deployment scripts** option

### Behavioral differences

- If you turn on the 'Ignore indexes' option, differences in indexes for memory-optimized tables are ignored if the only differences between the tables are the indexes. However, if there are other differences between the tables, all indexes will be deployed.
- The 'Ignore collations' option is not applied to memory-optimized tables.

### Known issues and further improvements

We're currently working on adding support for:

- deploying memory-optimized tables from a SQL Server 2014 database to a database on an older version of SQL Server
- deploying a disk-based table from a database on an older version of SQL Server to a SQL Server 2014 database where there is a memory-optimized table with the same name
- dropping and recreating database level triggers for CREATE TABLE and DROP TABLE statements on a target database when memory-optimized tables are deployed
- deployment using transactions for other objects types when they're deployed alongside memory-optimized tables  
For example, if you deploy functions alongside memory-optimized tables, there will be no transaction block for the section in the deployment script for deploying the functions
- other new features in SQL Server 2014 including memory-optimized table types, natively compiled stored procedures, clustered columnstore indexes and incremental statistics  
This latest SQL Compare beta may display incorrect SQL for these objects, and attempts to deploy them are likely to fail or result in incorrect deployment.

SQL Compare 11 won't include support for comparing and deploying backup files (native or SQL Backup) containing memory-optimized tables.

---

This version currently doesn't support:

- deploying memory-optimized tables from a SQL Server 2014 database to a database on an older version of SQL Server
- deploying a disk-based table from a database on an older version of SQL Server to a SQL Server 2014 database where there is a memory-optimized table with the same name
- comparison and deployment of backup files (native or SQL Backup) containing memory-optimized tables
- dropping and re-creating database level triggers for CREATE TABLE and DROP TABLE on a target database when memory-optimized tables are deployed
- deployment using transactions for objects types other than memory-optimized tables when they are deployed alongside memory-optimized tables. For example, if you deploy functions alongside memory-optimized tables, there will be no transaction block for the section in the deployment script for deploying the functions

Other new features in SQL Server 2014 are not currently supported, including memory-optimized table types, natively compiled stored procedures, clustered columnstore indexes and incremental statistics. SQL Compare may display incorrect SQL for these objects, and attempts to deploy them are likely to fail or result in incorrect deployment.

### Bug fixes

This version includes the following bug fixes:

---

SC-7260	Fillfactor option no longer appears in user-defined table types
SC-5164	Fix for the issue where SQL Compare attempts to re-add extended properties when altering a function with existing extended properties
SC-7248	Assemblies with circular dependencies on other assemblies no longer causes stackoverflow exception
SC-5443	Using drop and create instead of alter option no longer throws an error after dropping a view then trying to refresh it
SC-7142	Migration scripts are now retrieved by command line
SC-6835	*= operator in stored procedures is now properly parsed
SC-7063	A TIMEOUT value of -1 when retrieving messages from a queue no longer causes parsing error