

SQL Compare 7

July 2008

Note: these pages apply to a version of this product that is not the current released version.

For the latest support documentation, please see <http://documentation.red-gate.com>

Contents

Getting started	3
Worked example: Comparing and synchronizing two databases	4
Worked example: Using SQL scripts as a data source	14
Setting up the synchronization.....	25
Using the Synchronization wizard	28
Warnings	35
Understanding the synchronization.....	38
Exporting data sources	42
Working with scripts folders	44
Working with snapshots	46
Working with backups.....	47
Check for updates	49
Comparing databases on unconnected SQL Servers.....	50
Comparing databases on different SQL Server versions.....	51
Creating a rollback script	53
Copying the structure of a database	54
Common error messages	55
Rollback on script failure or cancellation.....	58
Getting started with the SQL Compare command line	59
Working with command line interfaces.....	60
Integrating the command line with applications	63
Command line syntax	75
Getting started with SQL Changeset	94
Getting the latest database	96
Saving modifications to source control	104
Registering working folders	108
Committing changes.....	111
Troubleshooting SQL Changeset.....	113
Acknowledgements	115

Getting started

SQL Compare enables you to compare and synchronize 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 migration scripts. SQL Compare automates this process for you.

You can use SQL Compare to compare and synchronize SQL Server 2008, SQL Server 2005 and SQL Server 2000 databases.

SQL Compare: step-by-step

1. Create or edit a comparison project
2. Choose the data sources to compare
3. View the comparison results
4. Set up the synchronization
5. Create the synchronization script

Worked examples

Learn more about SQL Compare by following these detailed examples:

- Comparing and synchronizing two databases
- Using SQL scripts as a data source

Note that you can follow the second worked example only if you are using SQL Compare Professional edition.

Related products

SQL Changeset is provided with SQL Compare Professional Edition for integration with your source control system. For more information, see: Getting started with SQL Changeset (page 94).

Worked example: Comparing and synchronizing two databases

This worked example demonstrates a basic comparison and synchronization 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 Web site. As part of this upgrade, they have made a number of changes to the structure of the database. They have already transferred the changes from the development server to a staging server, but they now need to transfer the changes to the production server.

You can follow the example on your own system. You will need access to a SQL Server to do this.

Setting 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:

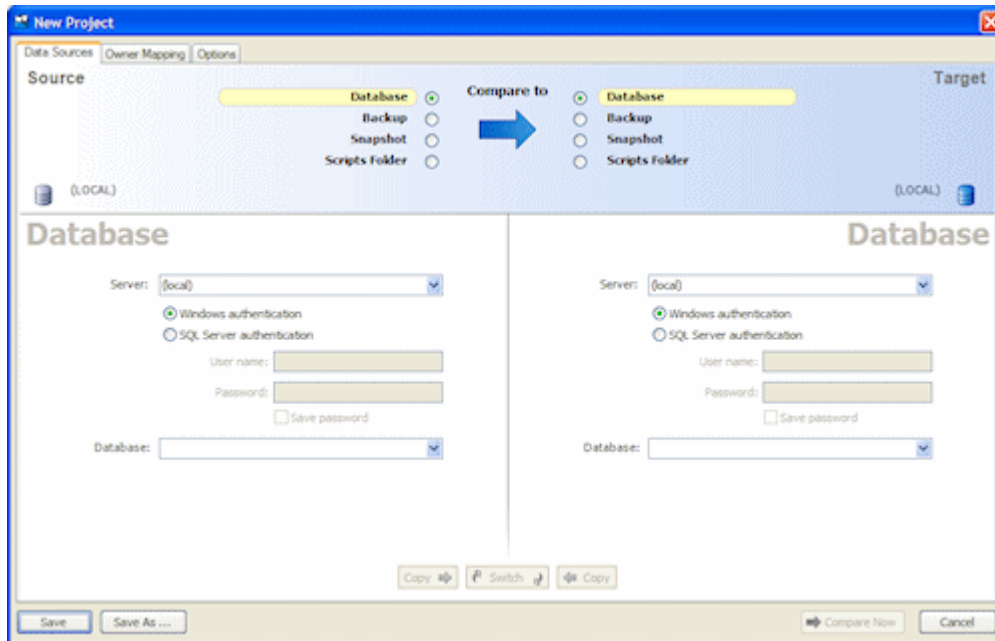
1. If they already exist, delete the databases *WidgetStaging* and *WidgetProduction* from your SQL Server.
2. Click here (/support/SQL_Compare/help/8.50/SQLCompareExampleGS.sql) to view the SQL creation script for the databases.
3. Copy the script, paste it in your SQL application, and then run it.

The databases and their schema are created.

Setting up the comparison

1. If you have not yet started SQL Compare, select it from your **Start** menu; if it is already running, click  **Comparison Projects**.
2. On the **Comparison Projects** dialog box, click  **New**.

The **Project Configuration** dialog box is displayed.



3. Ensure that the **Data source type** is set to *Live database*.

In this example, we will compare live databases; you can also compare snapshots that have been created using SQL Compare and scripts folders containing object creation scripts.

4. Type or select *WidgetStaging* in **Database** on the left side and *WidgetProduction* in **Database** on the right side.

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**.

5. Click **Compare Now**.

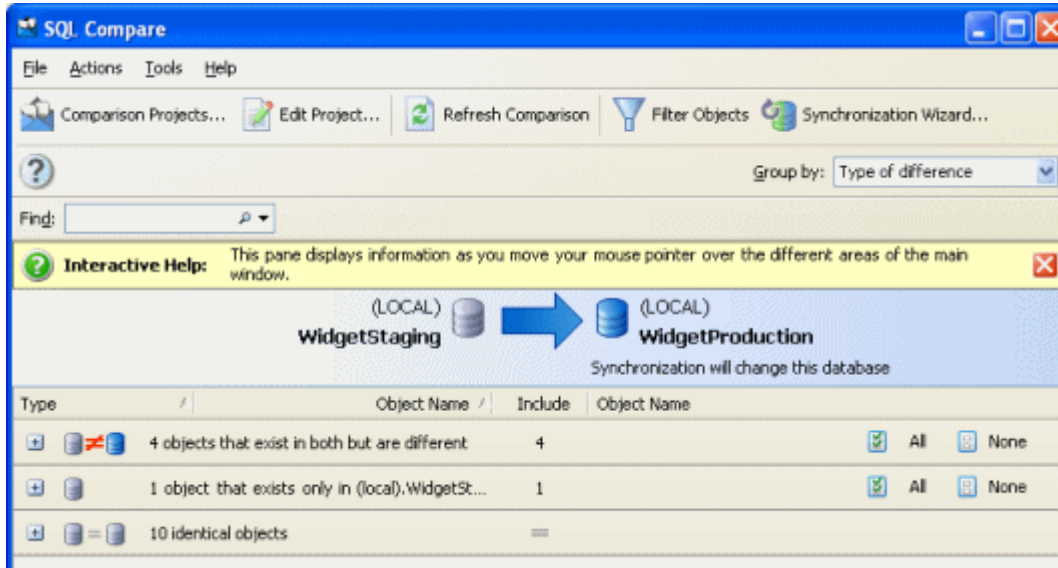
SQL Compare displays a message dialog box that shows the progress of the comparison.

If you select the **Close dialog box on completion** check box, SQL Compare closes this message dialog box automatically the next time that you run a comparison on a project. For this example, leave the setting as it is.




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


Viewing the comparison results





The comparison results are displayed in the main window.



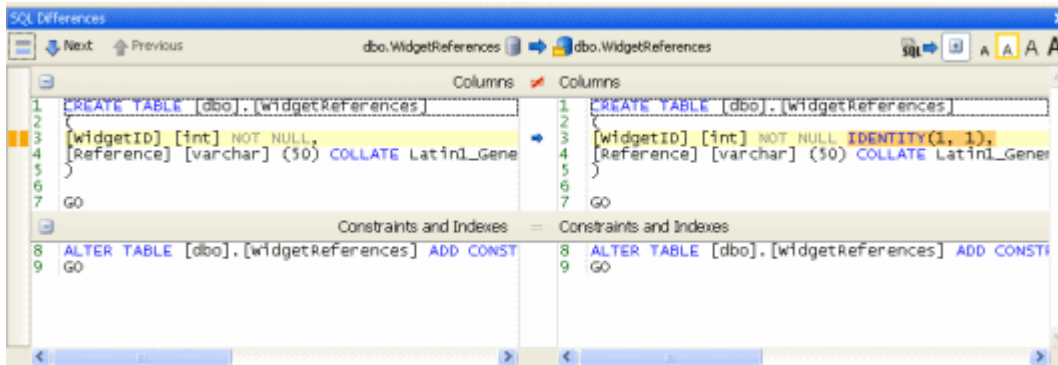
In this worked example, the comparison results are grouped by:

-  objects that exist in both databases but are different
-  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	Include	Object Name
4 objects that exist in both but are different	4		
Table	WidgetPrices	<input checked="" type="checkbox"/> 	WidgetPrices
Table	WidgetReferences	<input checked="" type="checkbox"/> 	WidgetReferences
Table	Widgets	<input checked="" type="checkbox"/> 	Widgets
View	CurrentPrices	<input checked="" type="checkbox"/> 	CurrentPrices
1 object that exists only in (local).WidgetStag...	1		
10 identical objects	=		

You can view a side-by-side, color-coded listing of the differences in the object creation SQL, by clicking an object. For example, if you click *WidgetReferences*, you can see the differences for the table *WidgetReferences*.




You can print a summary of the comparison results by opening the **File** menu and clicking **Print**. You can export the comparison results by opening the **Tools** menu and clicking **Export Comparison Results**.

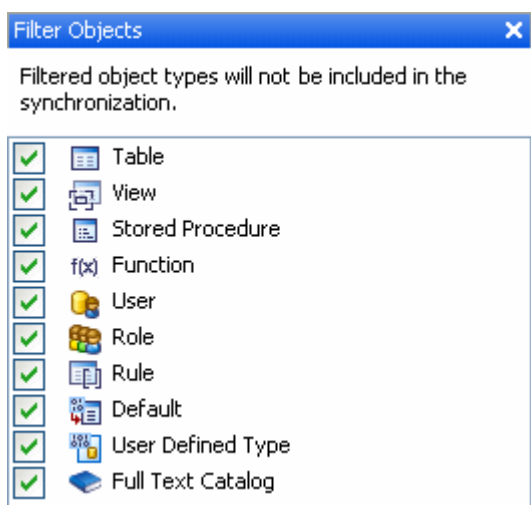
For full details of how to use the comparison results window, see [Viewing the comparison results](#).

Selecting the objects to synchronize

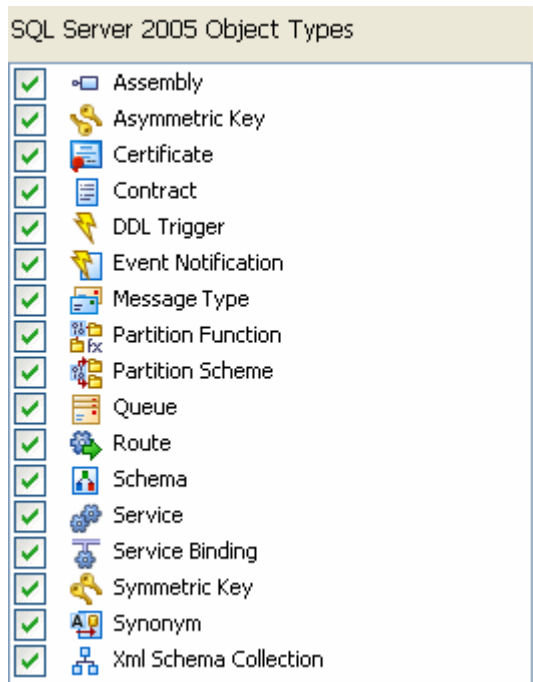
To synchronize the databases, you first select the objects you want to synchronize. You do this by using the appropriate check boxes in the **Include** column.

For this worked example, select all the objects that differ.

1. If the **Filter Objects** pane is not displayed, click  **Filter Objects**.





If you are using SQL Server 2005, a list of SQL Server 2005 object types is also displayed.



If you are using SQL Server 2008, a list of SQL Server 2008 object types is also displayed:



2. On the **Filter Objects** pane, ensure that all the object types are selected, so that all the objects are displayed in the main window, and are available for inclusion in the synchronization.
3. Click the close button  to close the **Filter Objects** pane.
4. For each of the object groups, ensure that all the objects are selected by clicking  **All**.

Note that you need to do this only for the following object groups:




objects that exist in both but are different

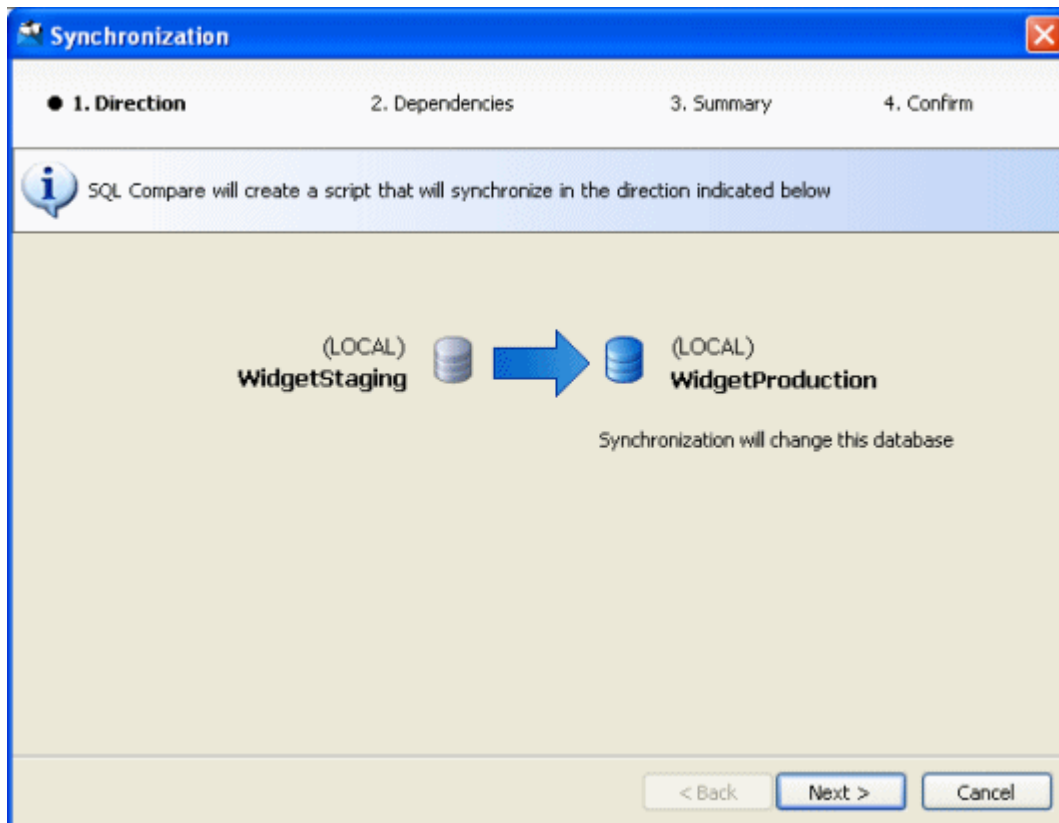


objects that exist only in (local).WidgetStaging

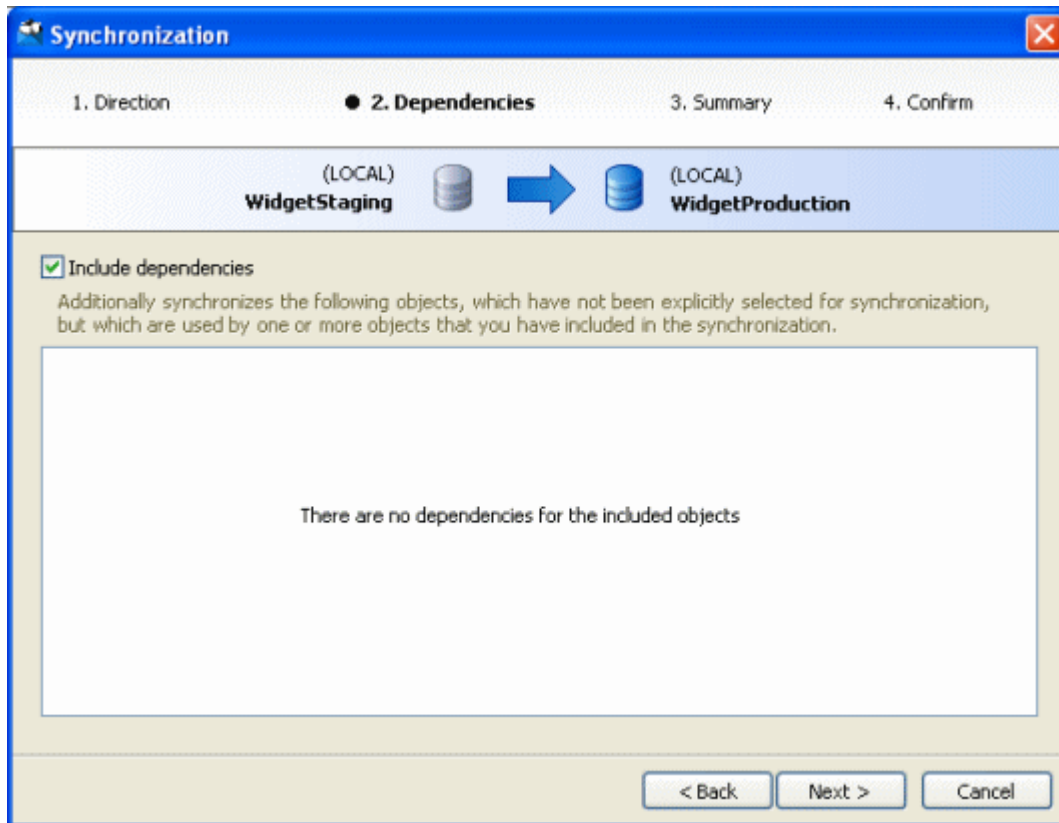
All the check boxes in the **Include** column are now selected.

Synchronizing the databases

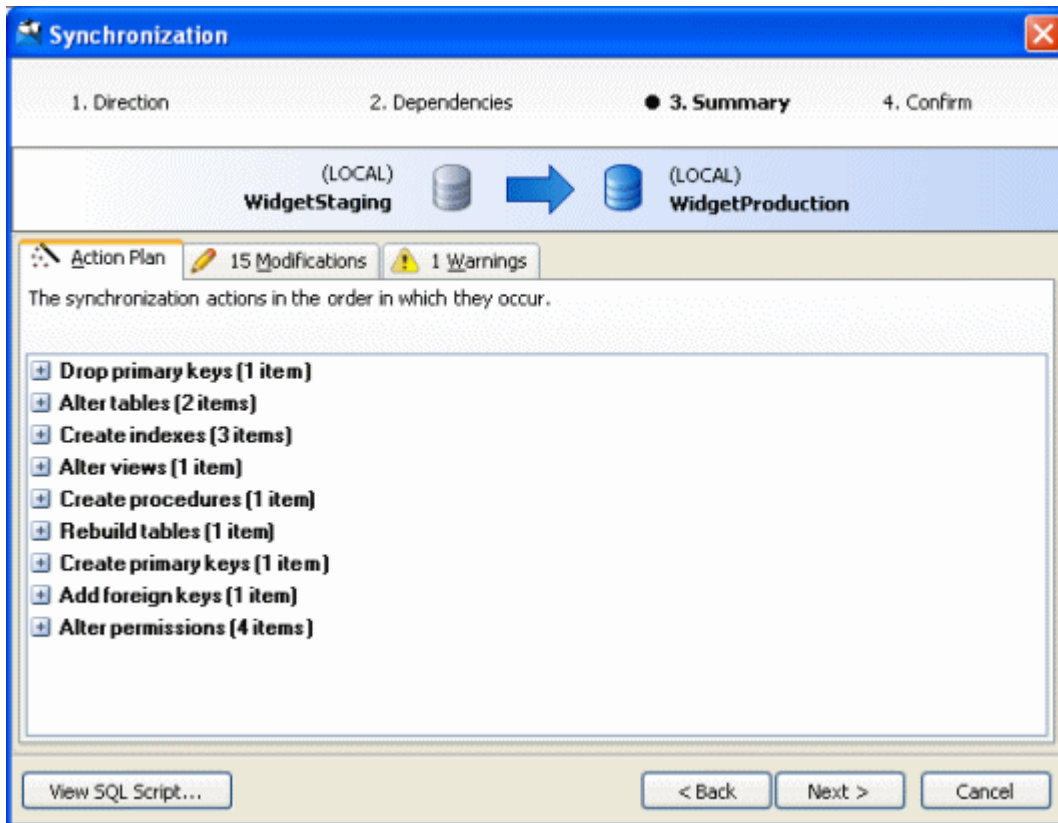
When you have selected the objects to synchronize, click  **Synchronization Wizard**.



Synchronization changes will be made to the WidgetProduction database. Confirm the direction in which the changes will be applied by clicking **Next**.



In this worked example, there are no dependencies. Click **Next** to generate the synchronization script.



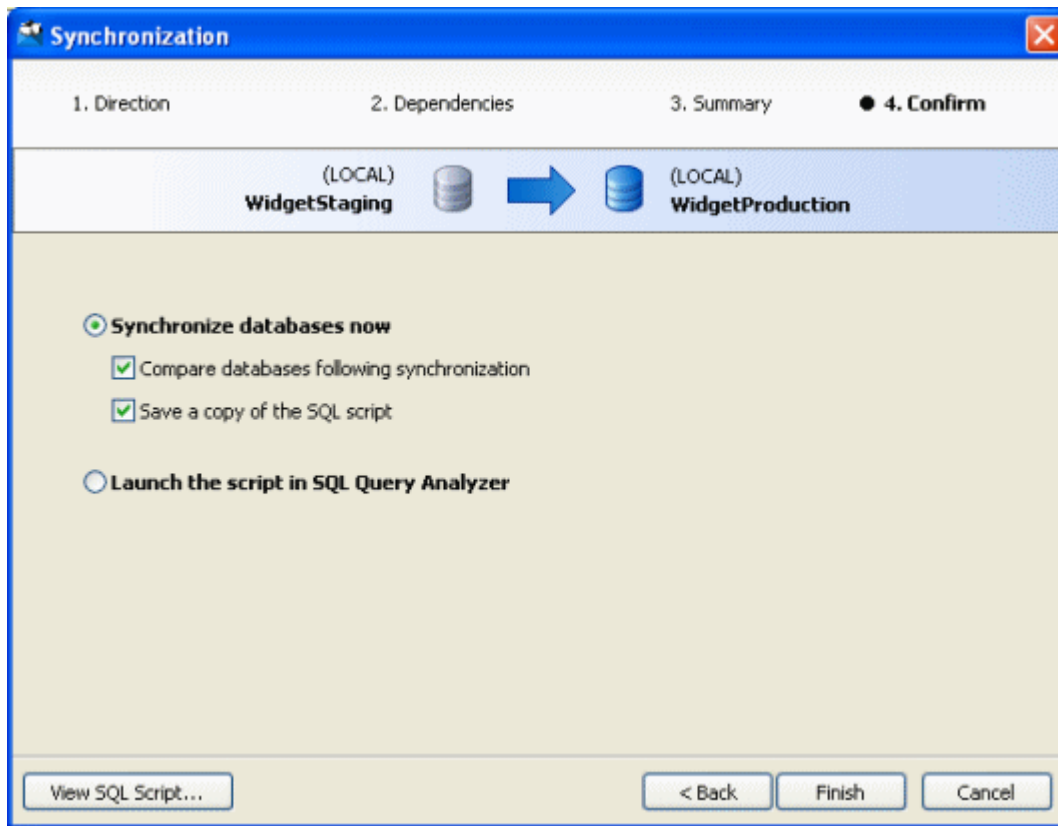
The **Summary** page displays the following tabs:

- **Action Plan** provides a synopsis of the script, grouped by command type, in the order in which the commands will run
- **Modifications** provides a synopsis of the script, grouped by object
- **Warnings** displays any warnings about inefficiencies in the script, or reasons the script may fail

In this example, SQL Compare displays a warning to inform you that it cannot use the ALTER TABLE command to change the IDENTITY column, so the synchronization 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.

You can display the synchronization SQL script by clicking **View SQL Script**. You can then save the script if required.

When you have looked at the script and warnings, click **Next** to go to the **Confirm** page.



You can choose either to run the SQL script from within SQL Compare or to launch your SQL application so that you can review the script.

In this example, we will choose to run the script, then compare the databases again to check the results, and save a copy of the synchronization script (the defaults).

Click **Finish** to run the synchronization.

A confirmation dialog box is displayed. Click **Synchronize Now** to continue.

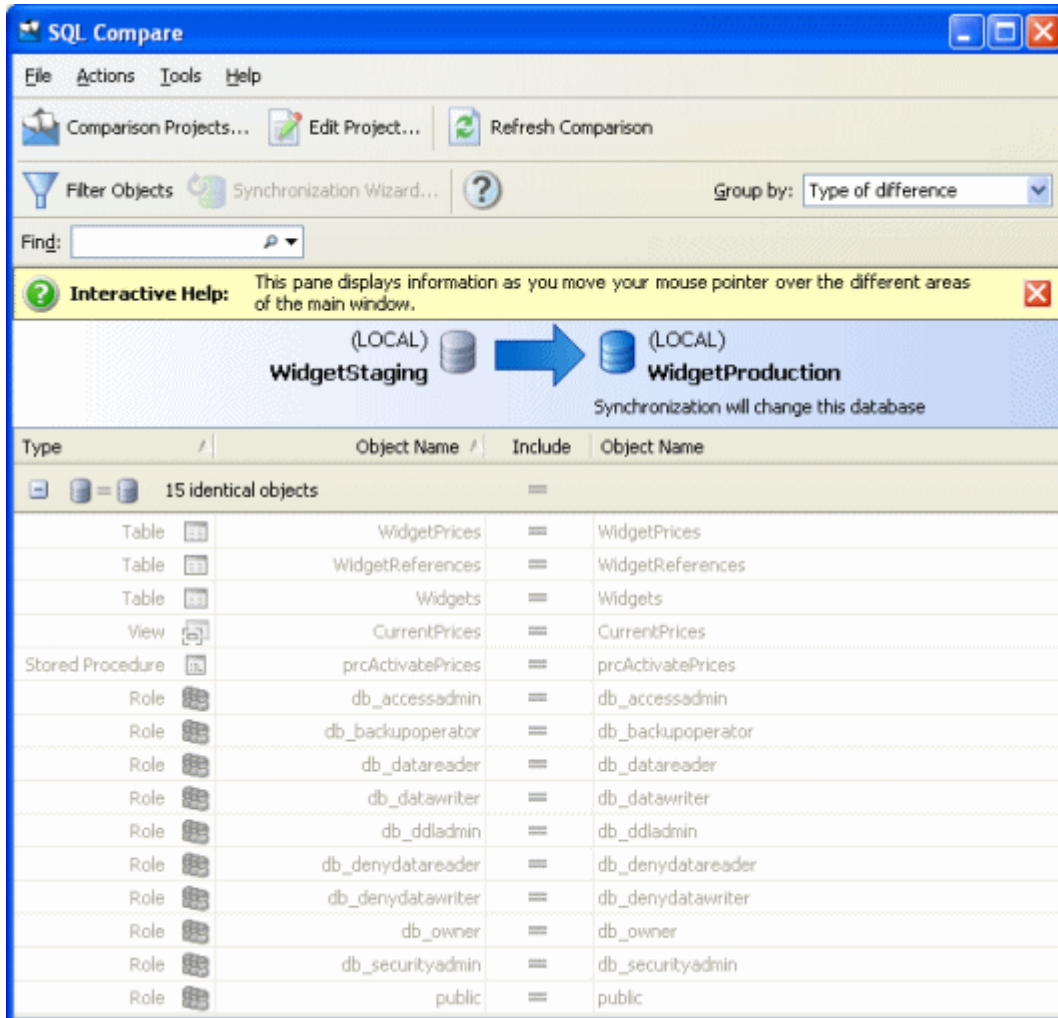
SQL Compare displays a message dialog box that shows the progress of the synchronization.

If you select the **Close message box on completion** check box, SQL Compare closes this message dialog box automatically the next time that you run a comparison on a project or synchronize databases. For this example, leave the setting as it is.

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

SQL Compare then re-compares the databases, and a message dialog box shows the progress of the comparison. Click **OK**.

The databases are compared and the results are shown in the main window. In this example, all objects are shown to be identical, confirming that the synchronization has been a success.



Worked example: Using SQL scripts as a data source

This worked example demonstrates the use of SQL object creation scripts as a database schema when comparing and synchronizing databases.

Note that if you are using a 'check out/edit/check in' (VSS style) source control system, you can use SQL Changeset to integrate SQL Compare Professional Edition with your source control system. However, this worked example assumes you are not using SQL Changeset. For examples and details of how to use SQL Changeset, see Using SQL Changeset (page 94).

In the example, the Super Sprocket Company has a SQL Server database running on a live server. This database 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, while ensuring that both the new and previous versions of the database structure are stored as creation scripts in a source control system. A copy of the production database has already been restored to an empty database, ready for development.

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 synchronizing two databases worked example, you are recommended to do so before starting this worked example.

Note that this worked example provides different databases for SQL Server 2000 and SQL Server 2005. The screenshots in this example use SQL Server 2005.

Setting 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
- SprocketStaging is a copy of the production database used for development

To create these databases on your SQL Server:


1. If they already exist, delete the databases **SprocketStaging**, **SprocketProduction**, and **SprocketDevelopment** from your SQL Server.
2. For SQL Server 2000 users, click here (/support/SQL_Compare/help/8.50/SQLCompareExampleScripts2000.sql) to view the SQL creation script for the databases.

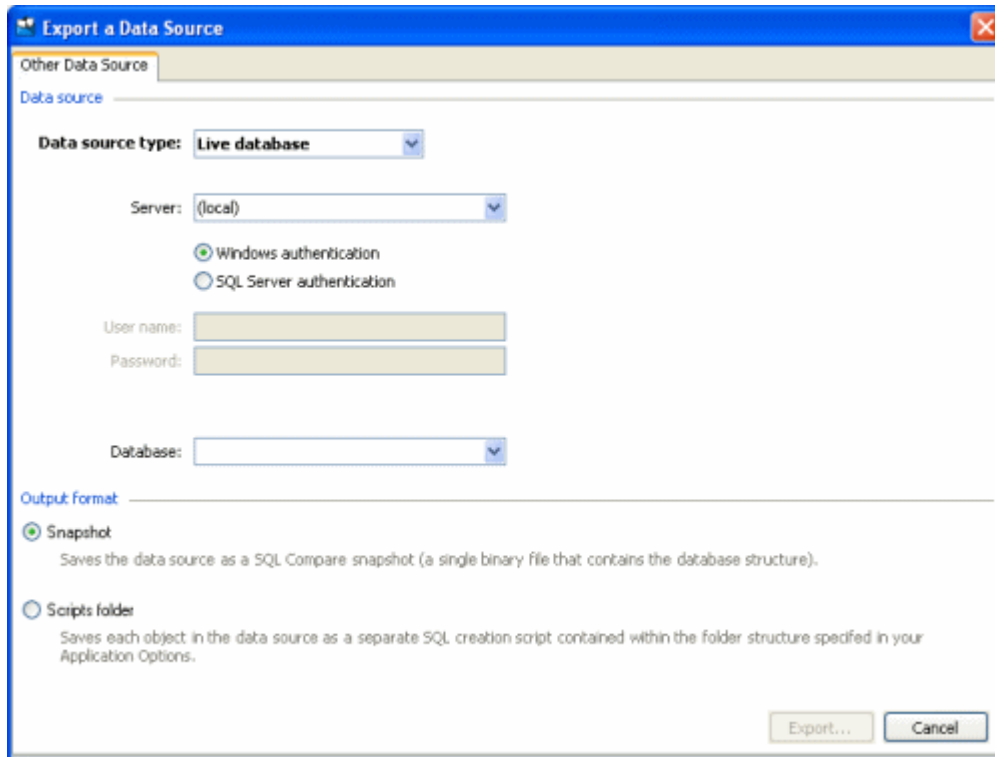
For SQL Server 2008 and 2005 users, click here (/support/SQL_Compare/help/8.50/SQLCompareExampleScripts2005.sql) to view the SQL creation script for the databases.

3. Copy the script, paste it in your SQL application, and then run it.
The databases and their schema are created.

Exporting the database to a scripts folder

To save the database schema, export it as a set of object creation SQL scripts that you can later use to recreate the schema, or to compare with another database.

1. If you have not yet started SQL Compare, select it from your **Start** menu; if it is already running, click  **Comparison Projects**.
2. On the **Comparison Projects** dialog box, click **Export a Data Source**. The **Export a Data Source** dialog box is displayed.



Select the **Other Data Source** tab, if it is not already displayed. If a comparison project was selected in the **Comparison Projects** dialog box, then the other tabs allow you to export the data sources in that project.

3. Ensure that the **Data source type** is set to *Live database*.

In this example, we will export a live database to a scripts folder; you can also export a SQL Compare snapshot to a scripts folder, or a scripts folder to a SQL Compare snapshot.

4. Type or select *SprocketStaging* in **Database**.

If the database is not displayed in the **Database** lists, right-click in the **Database** box and click **Refresh**, or scroll to the top of the list and click **Refresh**.

5. Under **Output format**, select **Scripts folder**.
6. Click **Export**.

The **Select a Folder** dialog box is displayed.

7. Browse to the location on your computer where you want to save the creation scripts. If you are using a source control system, this may be the folder designated as your working folder. To create a new folder, type the name in the **Folder name** box. In this example, call the folder *SprocketStaging*.
8. Click **Save**.

The schema is exported to the specified folder as a set of script files. Browse to the location on your computer where you saved the script files.

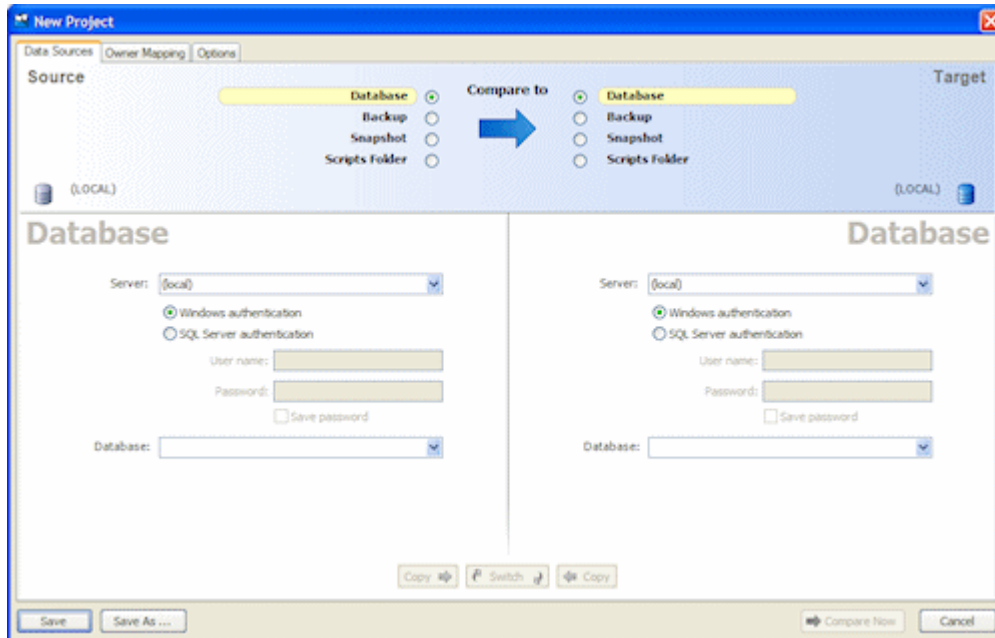
The script files are saved in a folder according to their object type; for example, all tables are saved in the Tables subfolder, and all views in the Views subfolder. You can change the subfolder used for each object type, if required. To do this, see Setting SQL Compare options (page **Error! Bookmark not defined.**).

Note that at this point in a development process, you may want to store the scripts in a source control system. This would allow you to roll back to this version of the schema at a later date, or to view changes to an individual object.

Comparing a database with the scripts folder

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. To compare the modified database with the schema saved as a scripts folder, set up a comparison project:

1. On the **Comparison Projects** dialog box, click  **New**.
The **Project Configuration** dialog box is displayed.



2. Set the **Data source type** on the left to *Live database*.
3. Type or select *SprocketDevelopment* in **Database** on the left side.
SprocketDevelopment is the new version of the database, following a development cycle.

- Set the **Data source type** on the right to *Scripts folder* and in the **Scripts folder** box, browse to the SprocketStaging folder you created. This will be the target data source in the comparison project.

SprocketStaging is the folder containing the stored version of the original database.

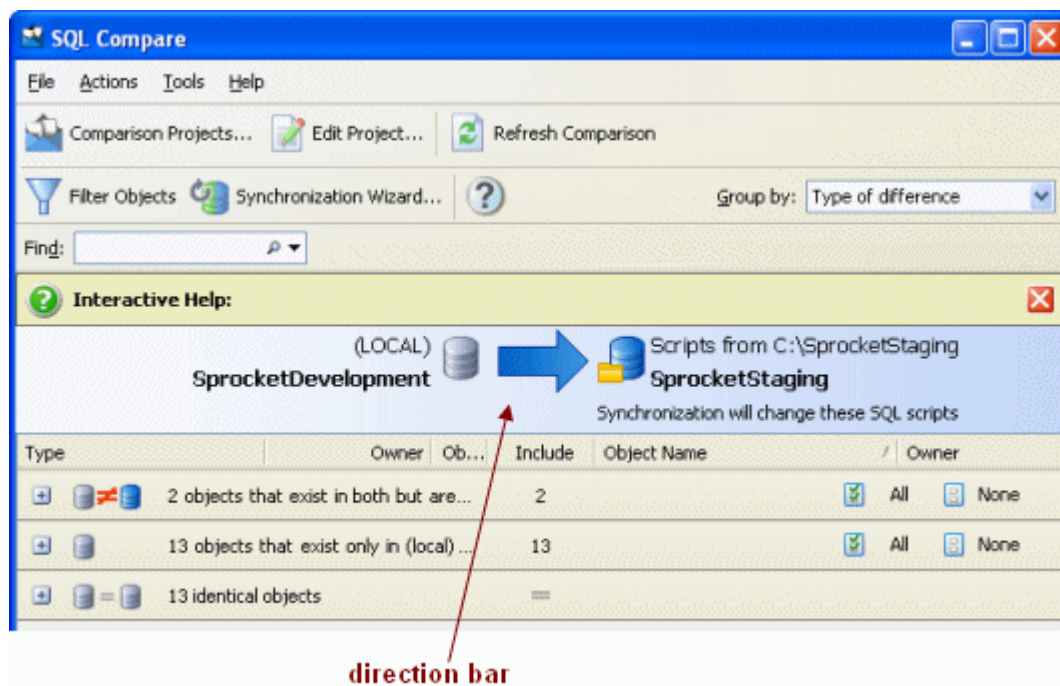
The collation and SQL Server version of the saved schema are displayed under **Database Settings**. For this example, leave the settings as they are.

- 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.

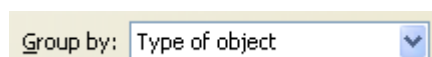
Viewing the comparison results

The comparison results are displayed in the main window.



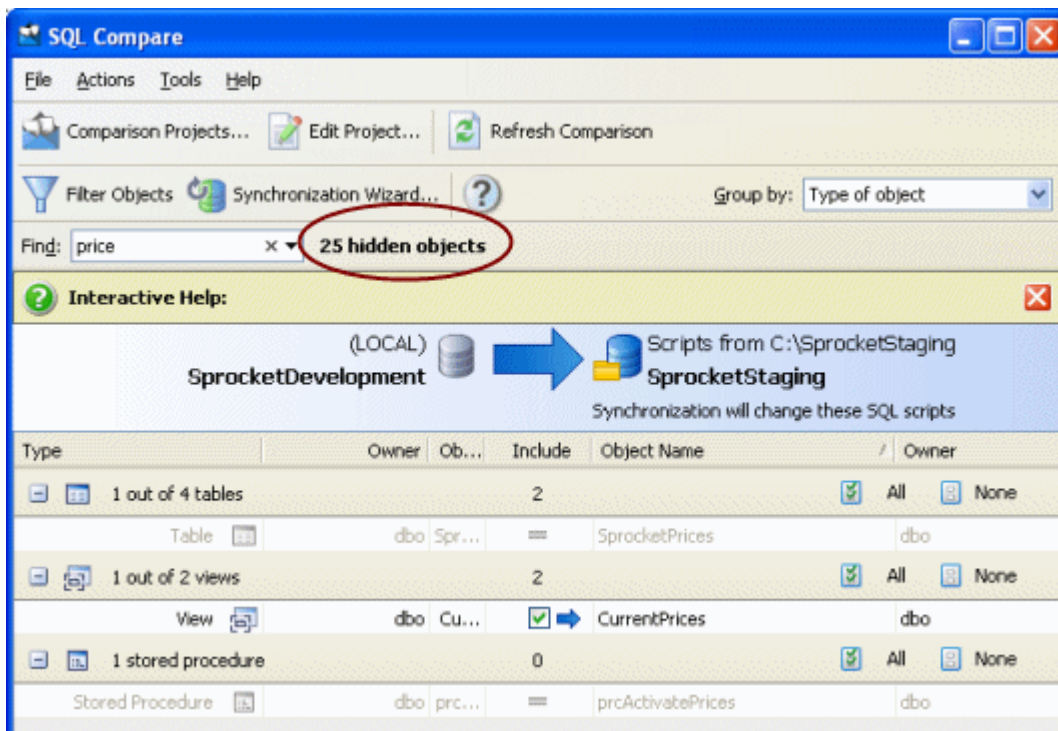
In the Direction bar, you can see that synchronization will change the SQL object creation scripts. The icon for the target data source  indicates that it is a scripts folder.

By default, the objects are grouped by type of difference. To group by object type, select *Type of Object* in the **Group by** box.



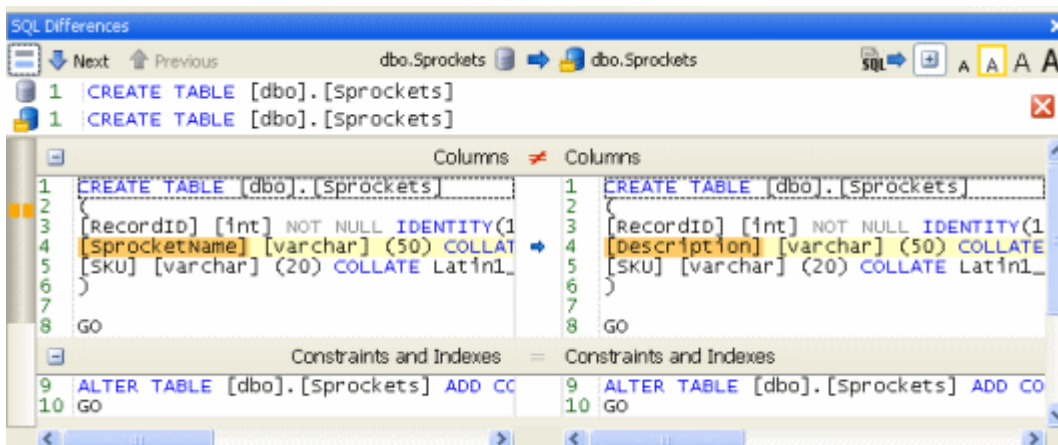
Right-click on a group and click **Expand all** to view all the objects in all the groups.




To search for objects, type the search text in the **Find** box. In this example, search for all objects that contain "price" by typing *price* in the **Find** box. SQL Compare searches object names and owner names.




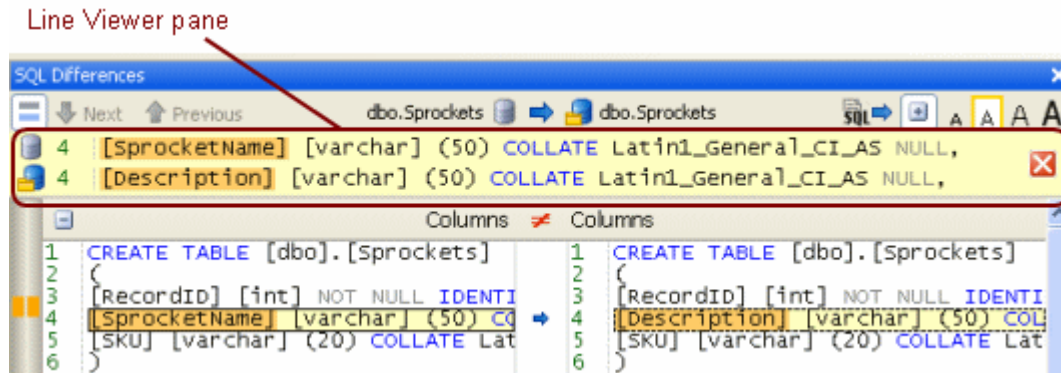
Some objects are now identified as *hidden*, because they do not match the search criteria. To clear the **Find** box, click the **x** button; all the objects are displayed again.

You can view a side-by-side, color-coded listing of the differences in the creation SQL, by clicking an object. For example, if you click the Sprockets table, you can see the differences for this table.



You can quickly go to lines that contain a difference using the  **Next** and  **Previous** buttons. Click  **Next** to go to the next line in the Sprockets table that is different; the

two versions of the line are shown one on top of the other in the Line Viewer pane. 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 Viewer. If the Line Viewer is not displayed, click  to display it.



For full details of how to use the comparison results window, see Viewing the comparison results.

Selecting the objects to synchronize

To save the modified schema, the object creation scripts need to be updated. Synchronizing the schemas will update the scripts, which can then be saved and labelled as the new version of the Super Sprocket Company database.

To synchronize the scripts, you first select the objects you want to synchronize using the appropriate check boxes in the **Include** column.

For this example, all objects will be synchronized.

Synchronizing the script files

When you have selected the objects to synchronize, click  **Synchronization Wizard**.

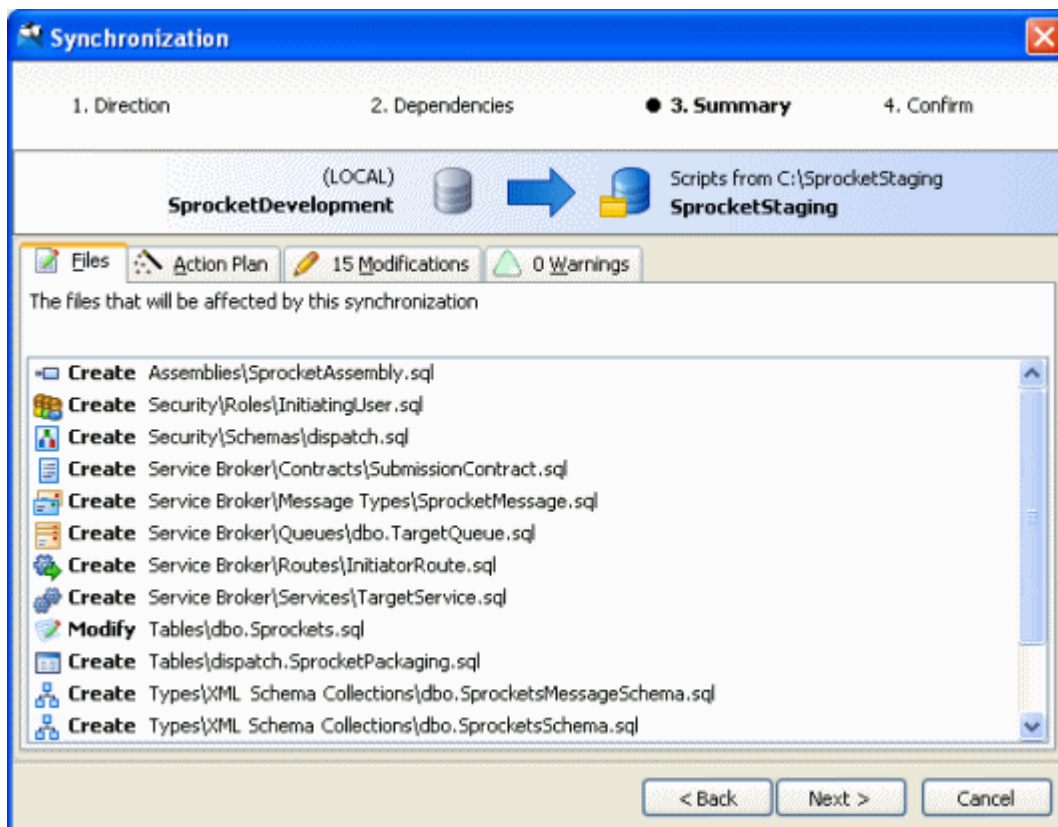
Synchronization changes will be made to the SQL creation scripts in the SprocketStaging folder. Click **Next** to view any dependencies.

In this worked example, there are no dependencies. Click **Next** to review the synchronization actions.

The **Summary** page displays the following tabs:

- **Files** lists the SQL creation scripts that will be modified or created when you synchronize. Note that files for objects that will be dropped during synchronization are shown as *Modify*. For more details, see *Working with scripts folders*.
- **Action Plan** provides a synopsis of the script, grouped by command type, in the order in which the commands will run
- **Modifications** provides a synopsis of the script, grouped by object
- **Warnings** displays any warnings about inefficiencies in the script, or reasons the script may fail

Click the **Files** tab to see the list of scripts that will be modified or created.



At this point in a development process, you may need to ensure that you have rights to access and edit the files that will be updated; when you synchronize, SQL Compare will warn you if read-only files need to be modified.

When you have looked at details of the changes to be made and viewed any warnings, click **Next** to go to the **Confirm** page.

Note that when you synchronize to a scripts folder, the SQL creation scripts in the folder are updated, but no synchronization script is produced. If you want SQL Compare to create a synchronization script, you can export the scripts folder to a snapshot and use the snapshot as the target data source. For more information, see *Working with scripts folders*.

In this example, the schemas will automatically be compared again once the synchronization is completed. If you do not want to re-compare the database schemas, clear the **Compare databases following synchronization** check box.

Click **Finish**.

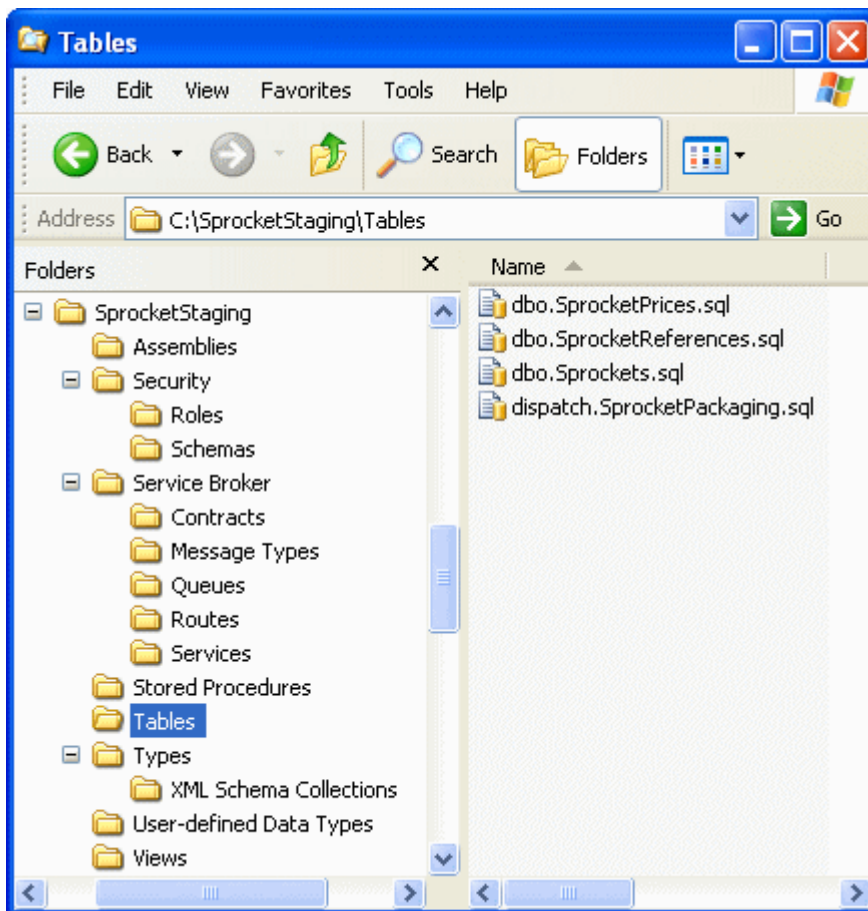
A confirmation dialog box is displayed. Click **Synchronize Now** to continue.

Click **OK** to close the dialog box.

SQL Compare then re-compares the databases, and a message dialog box shows the progress of the comparison. Click **OK**.

The databases are compared and the results are shown in the main window. In this example, all objects are shown to be identical, confirming that the synchronization has been a success.

Close the current comparison project and browse to the SprocketStaging folder you created earlier in the example.



There are now additional subfolders containing the new object types that have been created as a result of the synchronization. In this example, the default folder structure provided by SQL Compare is used to store the SQL creation scripts for each object type.

At this point in a development process, you could check these files back into your source control system. If you do this, you would now have both the previous saved version of the schema, and a set of updated files representing the new, modified version.

Using the script files to update a live database

You now have a stored set of script files containing the next version of the Super Sprocket Company database schema. You want to compare this new development version of the database with the current production database and synchronize the schemas.

Create a new comparison project as follows:

1. Set the **Data source type** on the left to *Scripts folder* and select *SprocketStaging* in the **Scripts Folder** box.

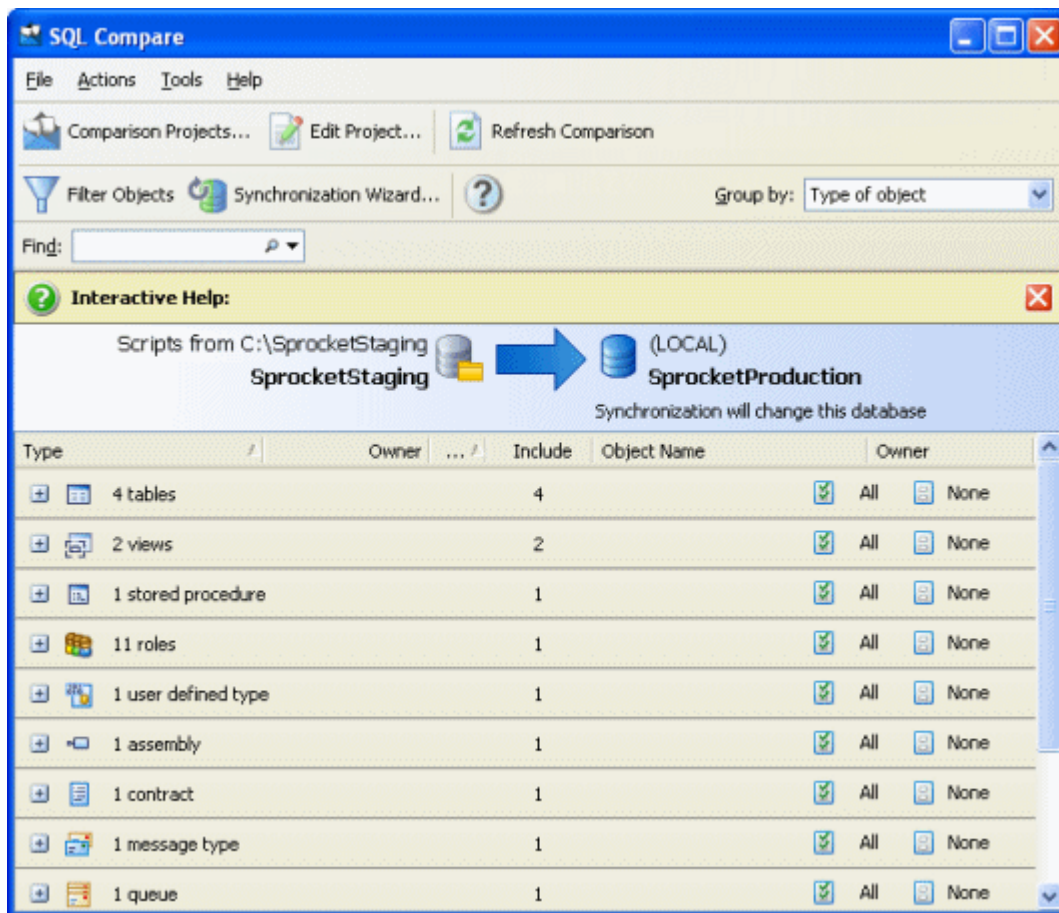
The SprocketStaging folder contains the new version of the schema. At this point in a development process, you may need to check out all the relevant files from your source control system.

2. Set the **Data source type** on the right to *Live database* and select *SprocketProduction* in the **Database** box.

SprocketProduction is the production database schema you want to update with the changes made during the development process.


3. Click **Compare Now**.

The comparison results are displayed in the main window.




Note that the comparison results are still grouped by *Type of object*. SQL Compare remembers the last setting you used.

To view the synchronization script for an individual object, click on a group heading to expand the group, then select the object in the comparison results; in the **SQL**

Differences pane, click  to view the script. To switch back to the creation script, click the button again.

Now that you have reviewed the changes, you want to update the production database schema. To do this:

1. When you have selected the objects to synchronize, click  **Synchronization Wizard**. In this example, we will synchronize all the objects (the default setting). Synchronization changes will be made to the SprocketProduction database.
2. Click **Next** to view the **Dependencies** page. There are no dependencies.
3. Click **Next** again to view the **Summary** page.

In this example, SQL Compare displays a warning to inform you that the SprocketReferences table will be rebuilt. Warnings are displayed whenever tables require rebuilding as these may be slow operations.

4. Click **Next** to view the **Confirm** page.

To save a copy of the synchronization script, ensure that **Save a copy of the SQL script** is selected. Saving the script means you can run it to synchronize other copies of the production database, to update them all.

5. Click **Finish**.


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

SQL Compare then re-compares the databases, and a message dialog box shows the progress of the comparison.

7. Click **OK**.

Setting up the synchronization

When you have reviewed the comparison results, you can run the Synchronization Wizard to create the SQL script to synchronize selected objects:



1. Select the objects that you want to synchronize.
2. Click  **Synchronization Wizard** to start the Synchronization wizard.

Selecting the objects to synchronize

By default, the first time that you run the comparison on a particular project, all objects are selected for synchronization. You can modify the selection by using the appropriate check box in the **Include** column in the comparison results.

Type	Object Name	Include	Object Name
4 objects that exist in both but are different		4	
Table	WidgetPrices	<input checked="" type="checkbox"/>	WidgetPrices
Table	WidgetReferences	<input checked="" type="checkbox"/>	WidgetReferences
Table	Widgets	<input checked="" type="checkbox"/>	Widgets
View	CurrentPrices	<input checked="" type="checkbox"/>	CurrentPrices


You can select multiple objects for synchronization by highlighting the objects using SHIFT + Click or CTRL + Click, and then clicking the **Include** check box for one of the highlighted objects.

To select all the objects in a group for synchronization, click  **All**; alternatively, you can right-click the object group and click **Include All Objects in this Group**. To exclude all objects displayed in an object group from the synchronization, click  **None**; or, you can right-click the object group and click **Exclude All Objects in this Group**.


The **Include All** and **Exclude All** options in the main **Actions** menu enable you to include or exclude all objects that are displayed in all object groups in the comparison results.

You can filter the object types and use the **Find** box to assist you with your selection as illustrated in the examples below.


Note the following:

- If you select the **Include** check box for an object and then you filter that object type, the object is not included in the synchronization; only selected objects that are shown in the comparison results are included. For example, if you select *TableA*, filter all tables, then click  **None** for *TableA*'s object group, *TableA* is still selected when you show the tables again.
- If you select the **Include** check box for an object and then run a search using the **Find** box and the object is hidden, then the object will still be included in the synchronization. Before synchronizing, you may want to clear the **Find** box.



Example: Selecting objects that exist only in the source database

1. Ensure that there are no filters on the comparison results so that all the objects are displayed.
2. Open the **Actions** menu and click **Exclude All** to clear all the check boxes.
3. From the **Group by** box, select *Type of difference* if it is not already selected. For the object group **objects that exist only in source database name**, click  **All** to select 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.
2. In the **Filter Objects** pane, right-click and click **Deselect All** to clear all of the object type filters, then select the  **Stored Procedure** check box to display only stored procedures in the comparison results.
3. Open the **Actions** menu and click **Include All**.




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.
2. In the **Filter Objects** pane, right-click and click **Deselect All** to clear all of the object type filters, then select the  **Table** check box to display only tables in the comparison results.
3. Open the **Actions** menu and click **Exclude All** to clear all the check boxes.
4. For the object group **objects that exist in both but are different**, click  **All** to select all of the tables in this group.

Example: Selecting only objects with a particular name

1. Open the **Actions** menu and click **Exclude All** to clear all the check boxes. This will ensure that no objects except those you specifically select will be synchronized.
2. Type the name of the object in the **Find** box.
Objects that match the search text are displayed.
3. For each group, click  **All** to select the visible objects only. If you have selected **No Groups**, select each object in turn for synchronization.

Example: Selecting tables and stored procedures with a particular name


1. Open the **Actions** menu and click **Exclude All**.
2. Clear all of the object type filters except for  **Table** and  **Stored Procedure** to display only tables and stored procedures in the comparison results.
3. Type the name in the **Find** box to display only objects that match the search string.
4. For each group, click  **All** to select the visible objects only. If you have selected **No Groups**, select each object in turn for inclusion.

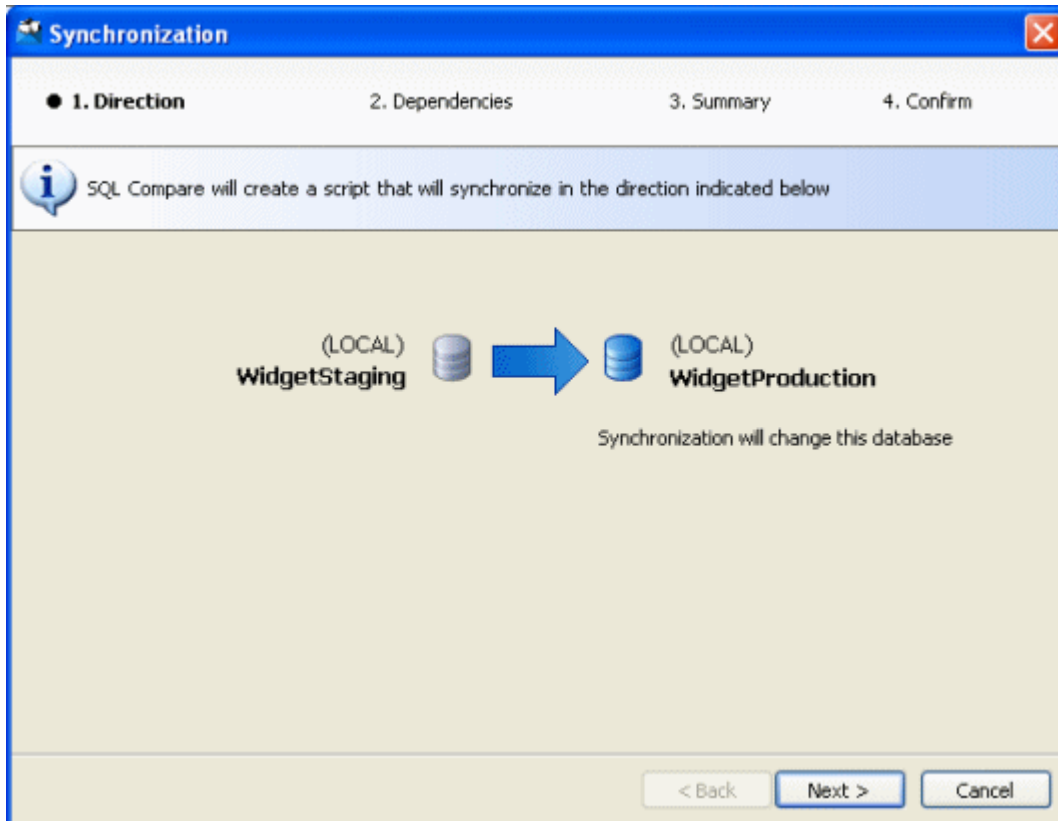
See also

Using the Synchronization wizard.....	28
Understanding the synchronization	38

Using the Synchronization wizard

When you have selected the objects that you want to include in the synchronization, you use the Synchronization Wizard to create the SQL script that will synchronize the databases.

To open the Synchronization Wizard, click  **Synchronization Wizard**.



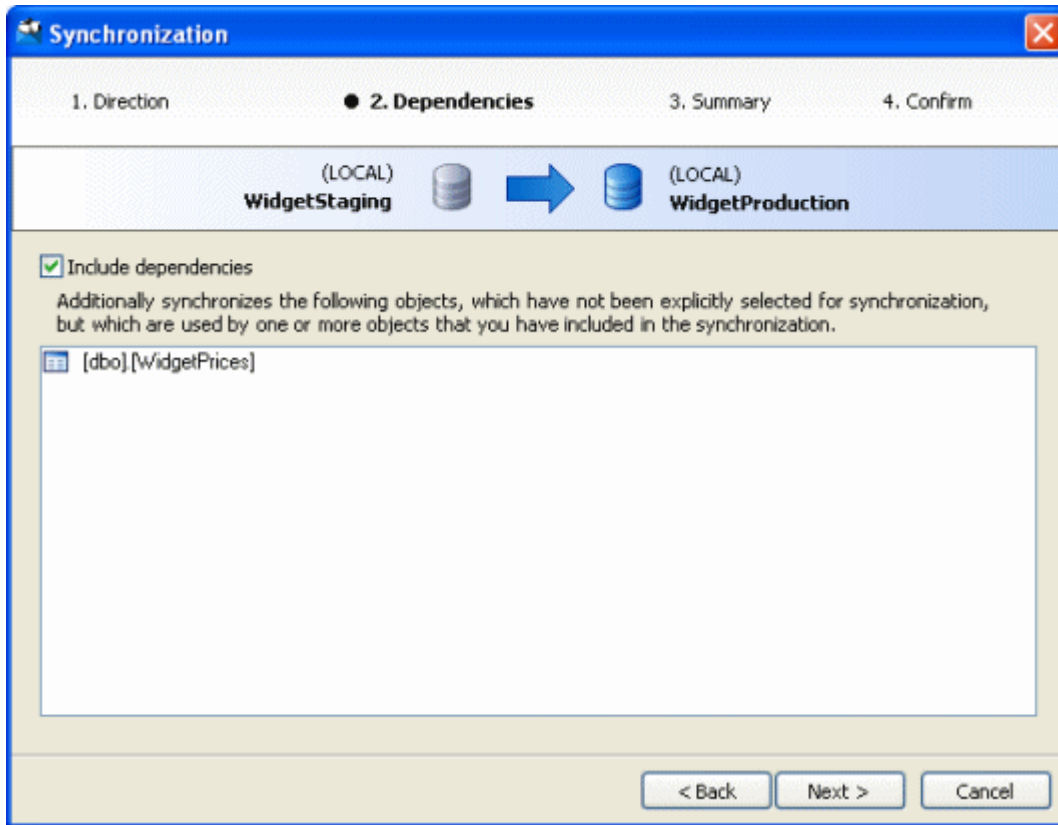
If you are synchronizing to a snapshot or a backup, note that you can generate the synchronization script, but you will not be able to run the script on the snapshot or backup. You may want to generate the synchronization script, for example, to create a rollback script.

Note that when you synchronize to a scripts folder, the SQL creation scripts in the target folder are updated, but no synchronization script is produced. If you want a synchronization script, you can export the scripts folder to a snapshot and use the snapshot as the target data source to generate the synchronization script.

Scripts folders are only available in the SQL Compare Professional edition.

Setting the direction and dependencies

You can change the direction in which the synchronization changes will be made by double-clicking the arrow. Click **Next** to confirm the direction in which the changes will be applied.



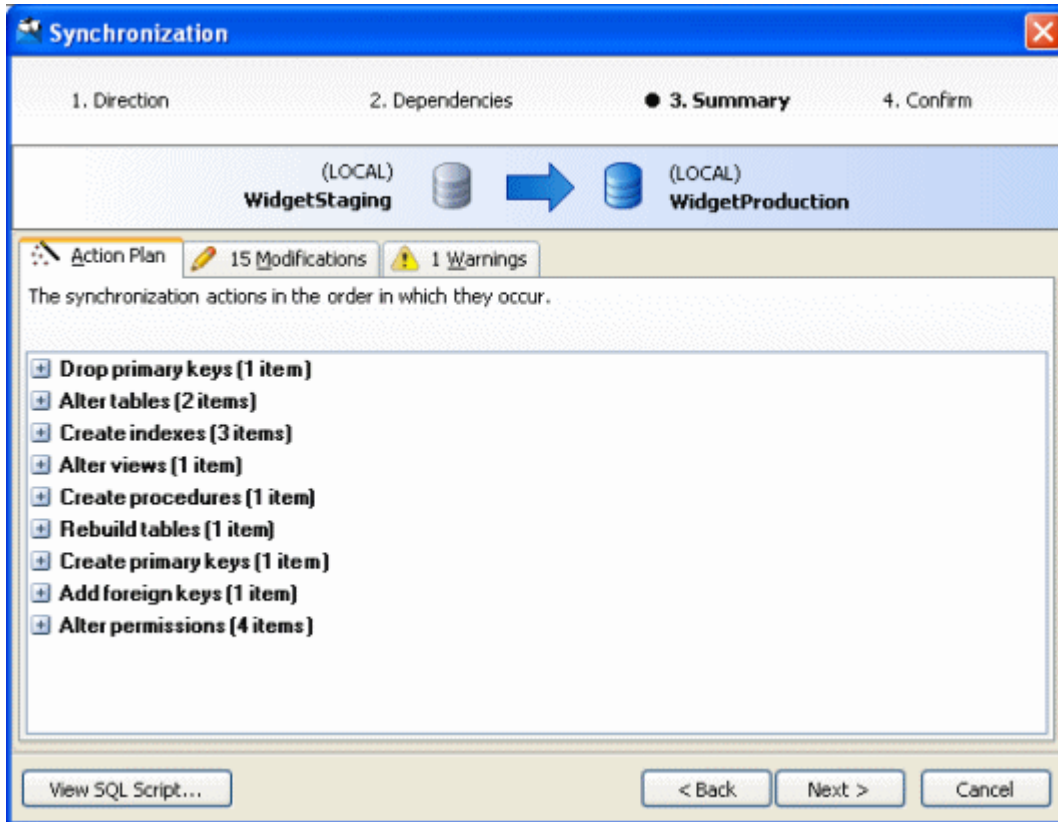
SQL Compare checks for object dependencies; if you excluded objects (including any filtered object types) from the synchronization and other objects that you selected are dependent on the excluded objects, those dependencies are listed. For example, if a stored procedure references a table, and you excluded that table from the synchronization, SQL Compare lists the table on the **Dependencies** page of the Synchronization Wizard.

By default, SQL Compare will include dependencies in the synchronization; clear the **Include dependencies** check box if you do not want to include the dependencies. Note that clearing this check box may produce unexpected results or the synchronization may fail.

Click **Next** to review the summary of synchronization actions.

Reviewing the synchronization summary

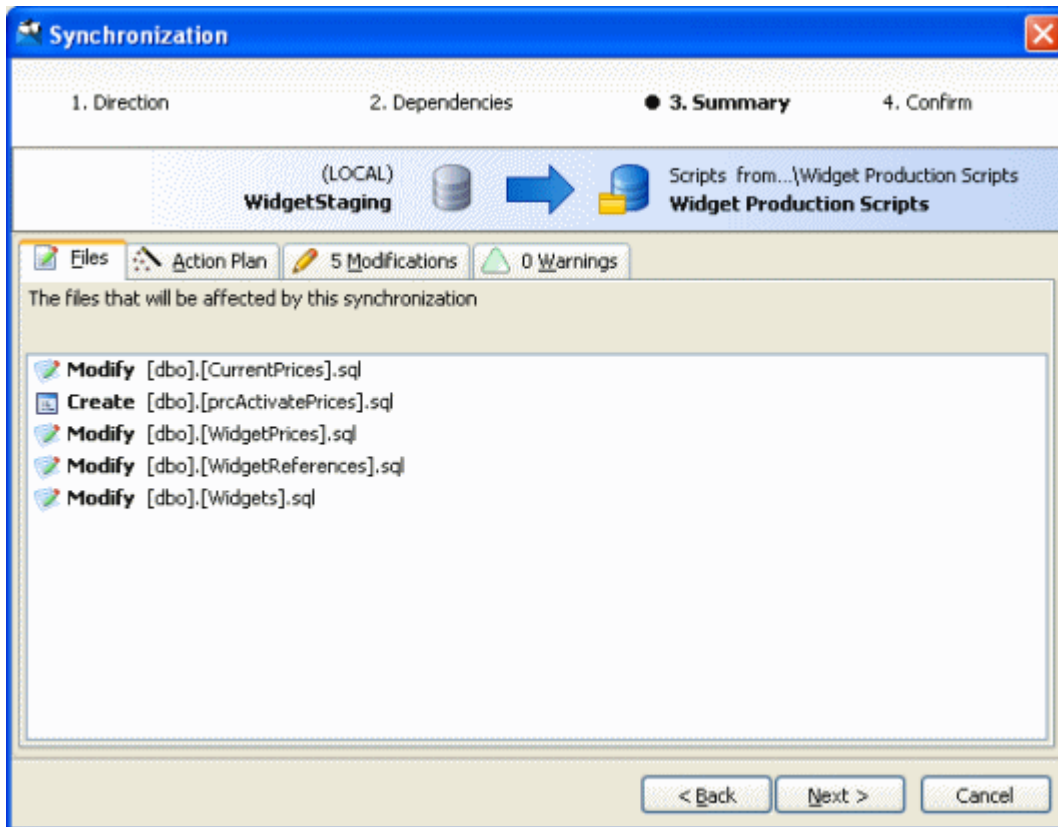
The **Summary** page lists each action or modification that will be carried out during the synchronization.



The **Summary** page displays the following tabs:

- **Action Plan** provides a synopsis of the script, grouped by command type, in the order in which the commands will run.
To view the commands for each command type, click **+**.
To copy the action plan, right-click and click **Copy Action Plan to Clipboard**.
- **Modifications** provides a synopsis of the script, grouped by object.
To copy the modifications, right-click and click **Copy Modifications to Clipboard**.
- **Warnings** displays any warnings about unexpected behavior that may occur when you synchronize the databases.
To copy the warnings, right-click and click **Copy Warnings to Clipboard**.

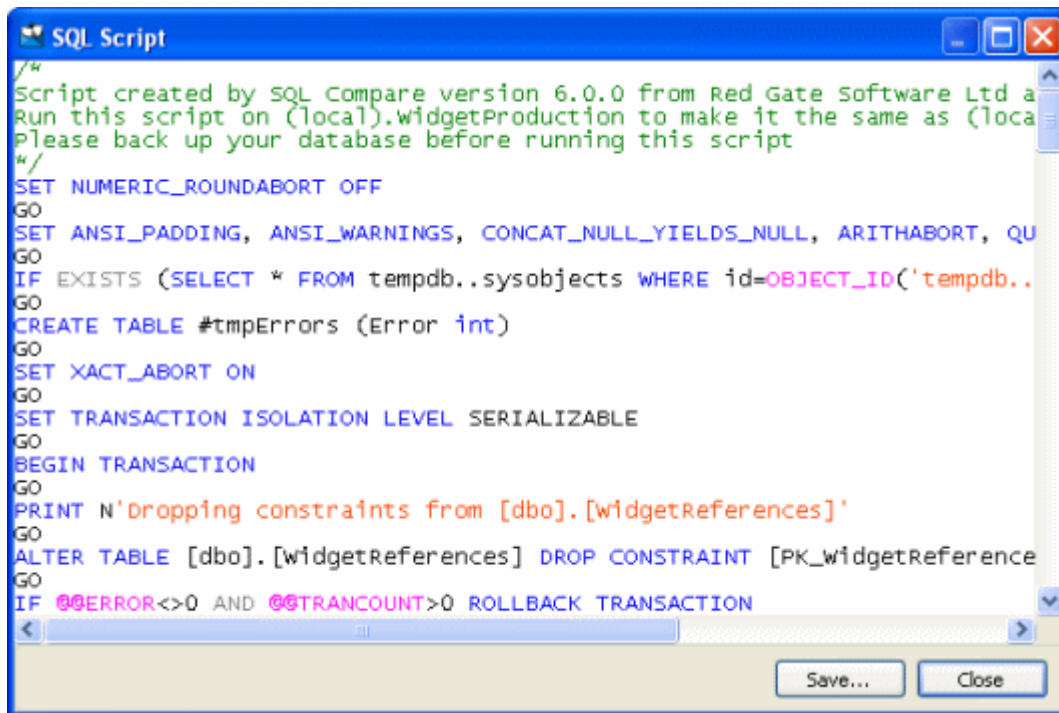
If you are synchronizing to a scripts folder, the **Summary** page of the Synchronization Wizard also displays a **Files** tab. The **Files** tab lists the object creation script files that will be modified or created during synchronization. To copy the file names, right-click and click **Copy File names to Clipboard**.



Note that when an object is dropped, its creation script file is not deleted; it will appear in the list as a **Modify** action. Before running the synchronization, SQL Compare will display a warning if any of the files that need to be modified are read-only, at which point you can choose not to synchronize. If you do proceed with the synchronization, the read-only files will be made writable and then modified. This may happen when you are working with a source control system that sets files to read-only status in some situations (if you are not using SQL Changeset (page 94) for source control integration).

Viewing the synchronization script



To display the synchronization SQL script, click **View SQL Script**.



```
SQL Script
/*
Script created by SQL Compare version 6.0.0 from Red Gate Software Ltd a
Run this script on (local).WidgetProduction to make it the same as (loca
Please back up your database before running this script
*/
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABORT, QU
GO
IF EXISTS (SELECT * FROM tempdb..sysobjects WHERE id=OBJECT_ID('tempdb..
GO
CREATE TABLE #tmpErrors (Error int)
GO
SET XACT_ABORT ON
GO
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
GO
BEGIN TRANSACTION
GO
PRINT N'Dropping constraints from [dbo].[widgetReferences]
GO
ALTER TABLE [dbo].[widgetReferences] DROP CONSTRAINT [PK_widgetReferenc
GO
IF @@ERROR<>0 AND @@TRANCOUNT>0 ROLLBACK TRANSACTION
```

This option is not available when synchronizing to a scripts folder.

You can:

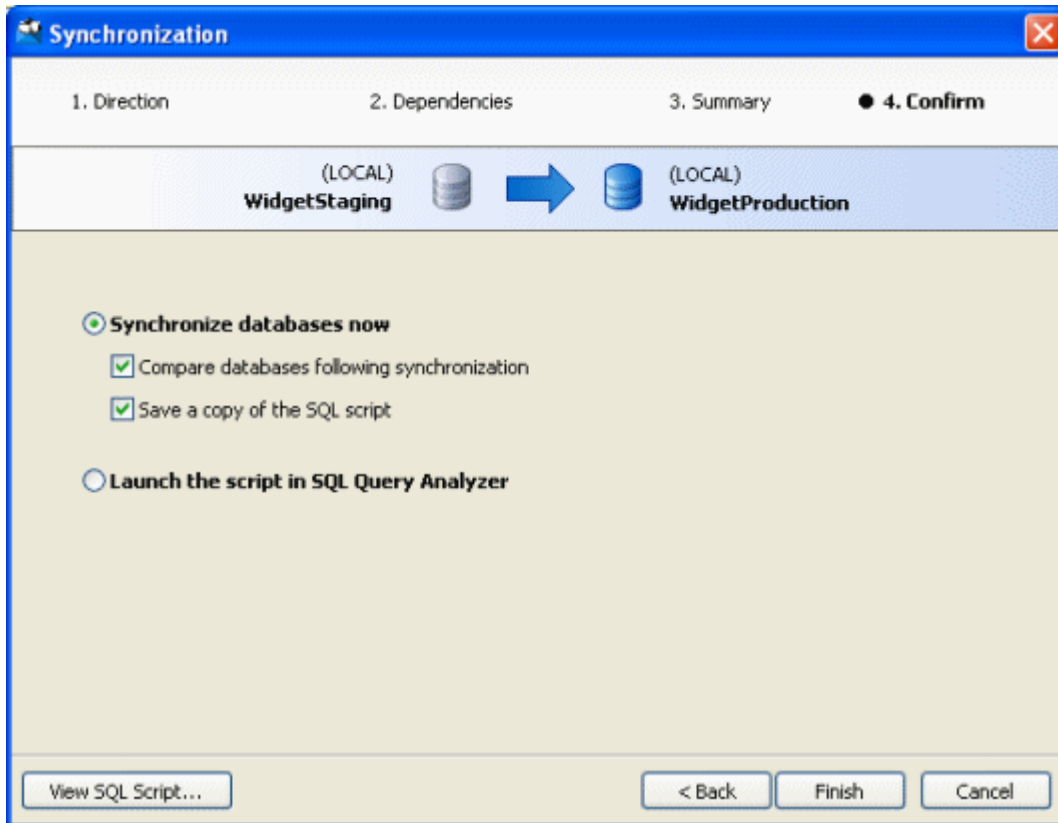
- search the script
Right-click and click **Find**; in the **Find** box, type the text to search for then click the  or  buttons to find the next or previous match.
- copy the script details for use in another application
Select the SQL statements, right-click, and then click **Copy**.
Alternatively, right-click, click **Select All**, then right-click, and click **Copy**.
- save the script to open it in a different application, or to keep a record of it
Click **Save**; a standard Windows® **Save As** dialog box is displayed.

When you have reviewed the script, click **Close**.

Click **Next** to specify how you want to use the synchronization SQL script.

Running the synchronization

The Confirm page allows you to set options for running the synchronization.



Note that if you are synchronizing to a scripts folder, you cannot save or launch the synchronization script. If you are using SQL Changeset to integrate SQL Compare Professional Edition with your source control system, you will see the **Check files out from source control before modifying** check box; for more information, see Saving modifications to source control (page 104).

It is recommended that you back up the database that is to be updated before you run the synchronization SQL script. Red Gate Software Ltd offers **SQL Backup** (http://www.red-gate.com/products/SQL_Backup/index.htm), which creates compressed, and if required, encrypted backups.

Do one of the following:

- Select **Synchronize databases now** to run the synchronization SQL script from within SQL Compare.
 - ♦ If you want SQL Compare to re-compare the databases on completion of the synchronization, select **Compare databases following synchronization**.
 - ♦ If you want SQL Compare to save the synchronization SQL script, select **Save a copy of the SQL script**. This option is not available if you are synchronizing to a scripts folder.
- Depending on your Application Options (page **Error! Bookmark not defined.**) settings, select **Launch the script in SQL Query Analyzer** or **Launch the script in**

SQL Server Management Studio to review the script. This option is not available if you are synchronizing to a scripts folder.

You can change the SQL application you use to open the script, and the location where the synchronization SQL script is saved by opening the **Tools** menu and clicking **Application Options**; you must close the Synchronization Wizard first.

Click **Finish**.

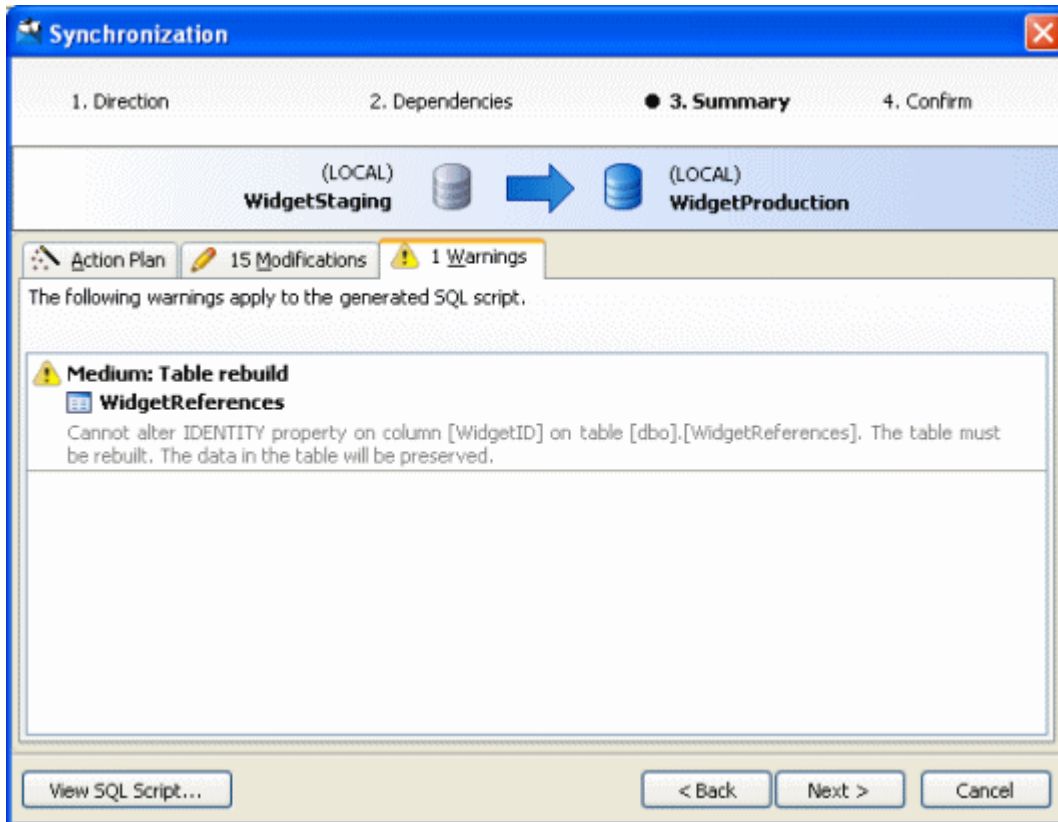
If you selected **Synchronize databases now**, a confirmation dialog box is displayed. Click **Synchronize Now** to start the synchronization.

If SQL Compare is unable to synchronize the data, an error dialog box is displayed, and where possible all changes are rolled back. Note that if you have selected the project option **Do not use transactions in synchronization SQL scripts**, the changes are not rolled back.

Note that if you are synchronizing to a scripts folder, and any of the script files that will be modified is designated as read-only, then a warning is displayed. If you click **Yes** to continue, then these files will be made writable so that they can be edited.

Warnings

When the Synchronization wizard displays the **Summary** page, you can click the **Warnings** tab to view any warnings about inefficiencies in the script, or reasons the script may fail. The warnings are graded according to severity.



You can copy the warnings so that you can paste them into another application by right-clicking the list of warnings and clicking **Copy Warnings to Clipboard**.

Some of the warnings that SQL Compare may display are summarized below. Further warnings concerning SQL Server 2005 and SQL Server 2008 databases may also be displayed.

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.

For example, 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 synchronization 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 synchronization. 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

The synchronization 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 synchronization 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 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 synchronization

This topic provides information that may help you to understand the behaviour of the synchronization script.

Column order

Column order is not forced unless you select the **Force column order** project option.

For example, your source database has a table that contains *Co/A* and *Co/B*, in that order, and your target database has the same table but with *Co/B* then *Co/A*. If **Force column order** is not 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 synchronization.

When column order is to be changed in a database, SQL Compare provides a warning in the **Summary** page of the Synchronization Wizard to notify you that the table will be rebuilt. SQL Compare uses temporary tables to ensure that any data in the table is not 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 *Co/A* in your source database is renamed to *Co/B* in your target database, when SQL Compare creates the synchronization SQL script, *Co/A* will be created in the target database as a new column and *Co/B* will be deleted. To avoid data loss, before you synchronize the databases you must take care to preserve any data in the two columns, and merge them following the synchronization.

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* does not 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 is not 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 synchronization 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.

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

Database diagrams

SQL Compare does not compare or synchronize database diagrams.

System objects

SQL Compare does not compare or synchronize system objects, except for users, roles, and system schemas.

Replication

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

Users

In Microsoft® Windows®, users are a composite of the domain name or computer name and the user name, for example *Computer1\WindowsUser1*. 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 synchronize users, ensure that their user names are different.

SQL Compare compares and synchronizes changes to users, such as changes to permissions. However, SQL Compare does not compare or synchronize modifications to user passwords.

Filegroups

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

When a filegroup is to be changed in a database, SQL Compare provides a warning in the **Synchronization script** page of the Synchronization Wizard to notify you that the table will be rebuilt. SQL Compare uses temporary tables to ensure that any data in the table is not lost.

Encrypted database objects

If you are synchronizing 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 synchronization SQL script. If you do not have system administrator privileges, you cannot synchronize 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 cannot compare the encrypted objects, or display their creation SQL, and cannot synchronize them.

SQL Compare version 7.0 (and earlier) cannot decrypt objects that are encrypted in a SQL Server 2005 or SQL Server 2008 database. If an encrypted object that cannot 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 cannot compare the encrypted objects, or display their creation SQL, and cannot synchronize 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. Note that the ToString representation of the CLR user-defined type must be the same for both the old and the new assembly, otherwise the synchronization 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 synchronizes the files. Note that 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. Note that if a CLR assembly synchronization also requires a table to be rebuilt twice, the CLR assembly and the partition scheme are synchronized 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 cannot compare all of the properties for a symmetric key.

If certificates, symmetric keys, and asymmetric keys are selected for synchronization, only the permissions are synchronized.

To ignore all certificates, symmetric keys, and asymmetric keys in the synchronization, 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 synchronized. If you do not want them to be synchronized, select the **Ignore extended properties** project option.

Numbered stored procedures

SQL Compare does not synchronize numbered stored procedures. However, you can synchronize them by running the synchronization SQL script in your SQL application.

Exporting data sources

SQL Compare enables you to export the structure of your database in the following ways:

- by creating a SQL Compare snapshot

SQL Compare snapshots are binary files that contain information about the structure of a database. They do not contain table data. Snapshots are useful for saving the structure of a database at a specific point in time; you can compare a schema with a snapshot to see what changes have been made.

- by creating a scripts folder

When you export a database schema to a scripts folder, SQL Compare generates a set of object creation script files contained within a folder structure that you specify. A script file is generated for each object in the database. You can export to a scripts folder only if you are using SQL Compare Professional edition.

Selecting the data source to export

You can export any type of data source: a live database, SQL Compare snapshot, backup, or scripts folder.

You can export the data source without having to re-enter the connection details. In the comparison results window, on the **File** menu, select **Export a Data Source**, then select the tab for the required data source.

To specify a data source that is not in a SQL Compare project, on the **File** menu, select **Export a Data Source**.

In the **Data source type** box, select one of the following:

- Live database


Type or select the name of the SQL Server in the **Server** box and then type or select the name of the database in the **Database** box.

For more information about entering connection details, see Setting data sources.


- Snapshot

Select the snapshot in the **Snapshot file name** box or click  to browse for the file.

- Scripts folder

Select the folder in the **Scripts Folder** box, or click  to browse for the folder.

- Backup

Click  to choose the backup files, then in the **Specify Files** dialog box, click **Add** to locate and add the required backup files. Select the required backup set using its **Compare** check box.

When the **Always use latest backup set** check box is selected (the default), SQL Compare automatically uses the latest backup set to export the database structure.

Creating a snapshot

To export the data source to a SQL Compare snapshot:

1. Under **Output format**, select **Snapshot**.
2. Click **Export**.
3. In the **Save As** dialog box, browse to the required folder, and then click **Save**.

Creating a scripts folder

When you export a data source to a scripts folder, the object creation scripts are saved in subfolders. To specify the subfolder for each object type, see Setting SQL Compare options (page **Error! Bookmark not defined.**).

To export the data source to a scripts folder:

1. Under **Output format**, select **Scripts folder**.
2. Click **Export**.
3. In the **Select a Folder for the Scripts** dialog box, browse to the required folder, and then click **Save**.

To create a new folder, type the folder name in the **Folder name** box. Note that if you select an existing folder, it must be empty.

Working with scripts folders

SQL Compare enables you to save a database schema as a set of object creation scripts. A separate script file is generated for each object in the database schema, and stored within a folder called a *scripts folder*. Different object types can be stored within subfolders that you specify. You can compare a scripts folder with a live database, a SQL Compare snapshot, or with another scripts folder.

You can specify a scripts folder as your target data source; when you run the synchronization, SQL Compare modifies or creates new object creation script files in the scripts folder. Note that when an object is dropped following synchronization, its script file is not deleted.

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.
Note that if you are using a 'check out/edit/check in' (VSS style) source control system, you can use SQL Changeset to integrate SQL Compare Professional Edition with your source control system. For examples and details of how to use SQL Changeset, see Using SQL Changeset (page 94).
- to compare databases on unconnected SQL Servers

You can also use your own object creation scripts in a comparison project. SQL Compare will determine any object dependencies and read the scripts in the correct order.

You can export to a scripts folder, or use a scripts folder in a comparison project, only if you are using SQL Compare Professional edition.

Creating, comparing, and synchronizing scripts folders

You can:

- export a database schema as a set of object creation scripts in a scripts folder
See Exporting data sources
- compare a scripts folder with another data source
See Setting data sources
- synchronize a scripts folder
See Using the Synchronization wizard

Generating synchronization scripts

SQL Compare does not generate a synchronization script when you synchronize to a scripts folder. If you want to generate a synchronization script, first export the target scripts folder as a SQL Compare snapshot. You can then use the snapshot in a comparison project and synchronize to generate the script.

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.
- **SQL Server 2005 and SQL Server 2008 encrypted objects**

Objects encrypted in SQL Server 2005 or SQL Server 2008 using WITH ENCRYPTION are not supported when synchronizing to or from a scripts folder.
- **Numbered stored procedures**

If you are using a scripts folder as a data source, numbered stored procedures are not supported.
- **CLR assemblies**

SQL Compare can compare object creation scripts that contain CLR assemblies, or include paths to assembly files. When synchronizing 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.
- **Mapping owners**

When synchronizing to or from script files, you cannot map owners.

Working with snapshots

SQL Compare enables you to save a snapshot of a database and compare it with a live database, or with another snapshot. A SQL Compare snapshot is a binary file containing information about the structure of a database; it does not contain any table data. You may want to save a snapshot of a database so that you can use it in a comparison project when you cannot connect to that database. For example, you can use snapshots:

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

Snapshots cannot be updated. However, if you compare a snapshot with a live database, the live database can be updated to synchronize it with the snapshot.

Creating and comparing snapshots

You can:

- save a database schema as a snapshot
See [Exporting data sources](#)
- compare a snapshot with another data source
See [Setting data sources](#)

Note that you cannot synchronize to a snapshot. Snapshot files cannot be updated.

Compatibility with previous versions of SQL Compare

Snapshots created using SQL Compare versions 3, 4 or 5 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 synchronization may fail when you use the snapshot in this version of SQL Compare.

Working with backups

SQL Compare enables you to compare backup files with a database, or with other backup files. This is useful, for example, when you want to retrieve the database structure from a backup file and then compare it with the structure of your database without running a restore operation or copying the backup file from a remote network. If you are comparing backup files with other backup files, you do not need SQL Server to be installed on your computer.

Note that comparing backup files is available only in SQL Compare Professional edition.

SQL Compare can retrieve the database structure from full or differential backups. However, you cannot retrieve the database structure from partial, filegroup, or transaction log backups.

When you run a comparison that uses backup files, SQL Compare locks the files when it reads them, and you cannot overwrite, move, or delete them.

Note that 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.

Comparing backups

You can:

- compare a backup with a database, snapshot, or scripts folder.
- compare a database, snapshot, or scripts folder with a backup. In this case, a synchronization script will be generated, but the backup is not altered.

For more information, see [Setting data sources](#).

Compatibility with backups

You can compare backups from SQL Server 2000, SQL Server 2005 and SQL Server 2008 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). Note that 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.

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

See also

Working with snapshots..... 46

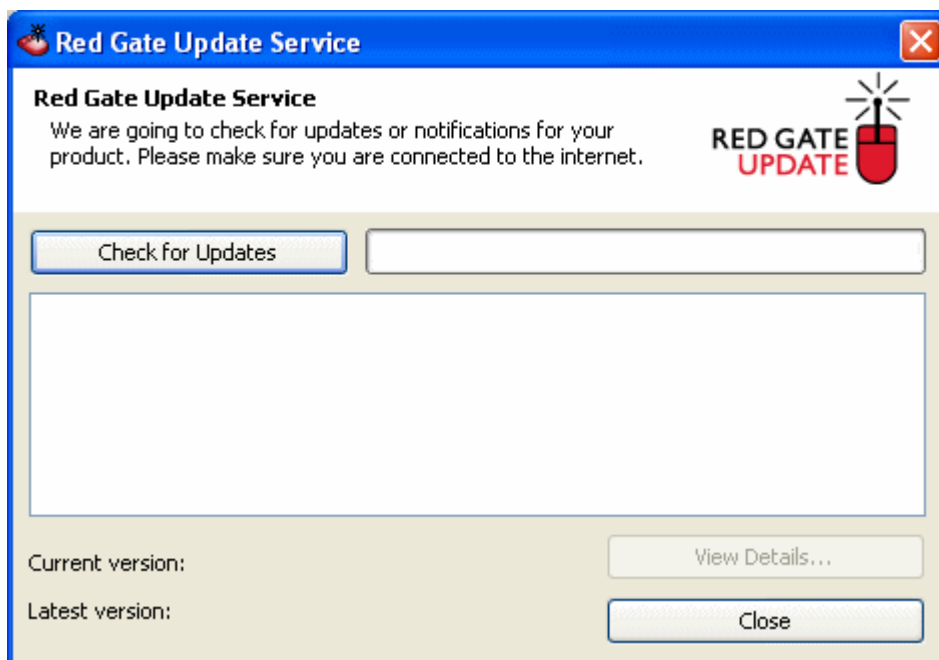
Check for updates

Periodically, Red Gate Software issues updates for SQL Compare. For example:

- free maintenance upgrades
- major upgrades that you can purchase
- notifications to inform you about new products or offers from Red Gate Software

To check for any updates that are available for download:

1. On the **Help** menu, click **Check for Updates**.



2. Ensure that you are connected to the Internet, and then on the **Red Gate Update Service** dialog box, click **Check for Updates**.

If your Internet connection uses a proxy server, ensure your Internet Explorer® connection settings are configured correctly. Note that **Check for Updates** does not work with automatic configuration scripts.

The updates that are available are listed.

3. Click **View Details**.

Red Gate Update Service displays the available updates and notifications in your default Internet browser for you to download.

Comparing databases on unconnected SQL Servers

You can compare databases on unconnected SQL Server instances by creating SQL Compare snapshots of the databases or a set of SQL scripts within a folder.

Using snapshots

To compare databases on unconnected SQL servers using snapshots:

1. Create a SQL Compare snapshot of the database(s). See Exporting data sources.
2. Copy the SQL Compare snapshot file(s) (.snp) to the required location.
3. Do one of the following:
 - ◆ Create a project that compares a SQL Compare snapshot with a database
 - ◆ Create a project that compares a SQL Compare snapshot with another SQL Compare snapshot
4. Run the comparison on the project.
5. Select the database objects that you want to include in the synchronization.
6. Use the Synchronization wizard to generate the synchronization SQL script.

You cannot use a synchronization SQL script to update a snapshot.

Using scripts folders

To compare databases on unconnected SQL servers using scripts folders:

1. Export the database schema to a scripts folder.
2. Copy the folder containing the SQL scripts to the required location.
3. Do one of the following:
 - ◆ Create a project that compares the scripts with a database
 - ◆ Create a project that compares the scripts with another set of scripts
4. Run the comparison on the project.
5. Select the database objects that you want to include in the synchronization.
6. Use the Synchronization wizard to update the database schema, object script files or generate a synchronization script.

You can export to a scripts folder, or use a scripts folder in a comparison project, only if you are using SQL Compare Professional edition.

Comparing databases on different SQL Server versions

This topic provides additional information for you if you are comparing databases that are on different versions of Microsoft® SQL Server™.

Comparing a SQL Server 2005 or 2008 database with a SQL Server 2000 database

If you are updating a SQL Server 2005 or SQL Server 2008 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 are updating a SQL Server 2000 database to match a SQL Server 2005 database, note the following:

- SQL Compare may be unable to synchronize all objects. Warnings will be displayed where possible.
- SQL Compare version 7.0 (and earlier) cannot decrypt objects that are encrypted in a SQL Server 2005 or SQL Server 2008 database. If an encrypted object which cannot 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 cannot compare the encrypted objects, or display their creation SQL, and cannot synchronize them.
- In SQL Compare version 7.1 (and later) you have the option to 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, clear the **Decrypt encrypted objects on 2005 and 2008 databases** check box.

When this option is disabled, SQL Compare cannot compare the encrypted objects, or display their creation SQL, and cannot synchronize them.

- An extra line of white space may be appended to the creation SQL 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 **SQL Differences** pane, and if these are the only differences, it shows the objects to be identical.

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 synchronizations 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 synchronizations may fail.

Creating a rollback script

You can create a rollback script by changing the synchronization direction and regenerating the synchronization SQL script.

For example, if you are updating Database B to match Database A, change the synchronization direction in the **Direction** page of the Synchronization Wizard, so that Database A is the database that will change.

To view the rollback script, click **View SQL Script** on the **Summary** page or the **Confirm** page of the Synchronization wizard; you can save the rollback script by clicking **Save**.

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. Synchronize the databases.

Alternatively, Red Gate Software Ltd 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.

Common error messages

Some of the more common error messages are explained below.

Could not start a transaction for OLE DB provider (name)

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.

SQL Server doesn't exist or access is denied

SQL Compare cannot 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.

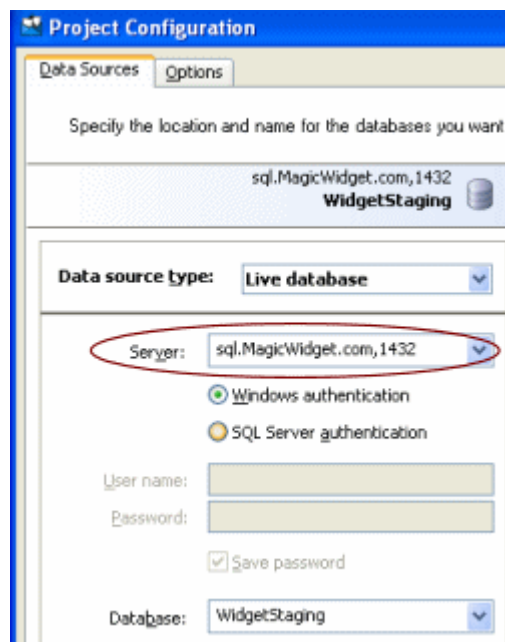
If the SQL Server is online, verify that you are connecting to the correct port.

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>
```

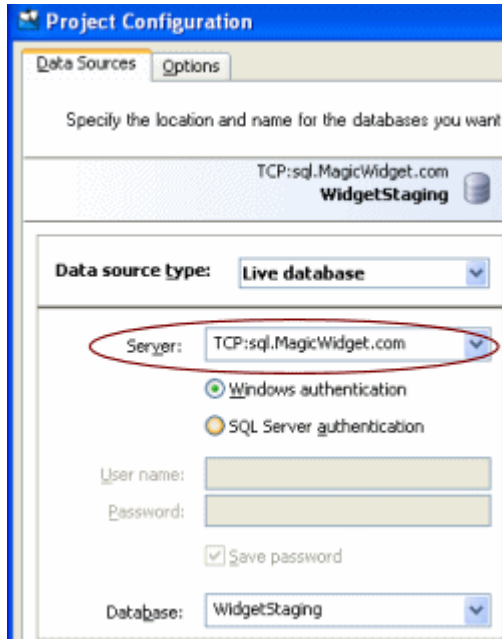
where *ServerName* is the name of your SQL Server and *Port* is the number of the port on which your SQL Server is running.

For example:



2. 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:



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; you must select this 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.

Could not enlist in a distributed transaction

SQL Compare may display this message if you are 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 synchronization, SQL Compare sets the transaction isolation level to SERIALIZABLE. To run the synchronization SQL script on the target database, you may need to change the transaction isolation level; open the script in your SQL application, and change the line:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

to:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```


Note that you should then take other precautions to ensure that no changes are made to the structure of the target database when you run the synchronization SQL script.

Rollback on script failure or cancellation

If a 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 in which this is not possible:

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

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

If you have selected the project option **Do not use transactions in synchronization SQL scripts** to remove transactions from the synchronization 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 is unable to roll back changes.

Getting started with the SQL Compare 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 synchronization of database schema
- perform scheduled comparisons and synchronizations
- synchronize 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#.

Prerequisites

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

- A SQL Compare Professional Edition, or SQL Toolbelt license
- If you do not have a license, you can use the command line for 14 days.
- .NET framework version 2.0 or later

This is required to run the command line interface, but it is not required when you develop applications and scripts that use the command line interface.

- MDAC 2.8 or later

Displaying help from the command line interface

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

```
sqlcompare /help /verbose
```

Learning more about the command line interface

For detailed examples using the command line see:

- Example: selecting single tables for comparison
- Example: selecting tables with unrelated names

Working with command line interfaces

This topic provides a description of basic command line features, as they are used with SQL Compare 7. For details of all of the switches that are available for the SQL Compare command line, see *Switches used in the command line*.

To display help on any of the tools from the SQL compare command line, enter:

```
sqlcompare /help
```

This displays a brief description of the tool, and basic help on all the command line switches.

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

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

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

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

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

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

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

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

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

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

```
/include:table:Product
```

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

Note that if you use the */include* switch to compare only tables (or any object type) matching a word or regular expression, all other objects not of that type will still be included in the comparison. In the above example, only tables that contain the word *Product* will be included, but all views, stored procedures, users and so on will still be included, unless further arguments to limit these object types are also specified.

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.

/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
```

However, note that if you set any options explicitly, all of the default options are switched off.

For more information about options, and a list of default option settings, see Project configuration 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 output

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

- Use the */out* switch to specify the file to which you want output directed:

```
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.

- Use the output redirection features that are provided by the shell in which you are executing the command.

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

```
sqlcompare ... > outputlog.txt
```

Note that 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 tool with applications that you distribute to your customers, you must first purchase either a valid SQL Compare Professional Edition, or SQL Toolbelt license. When you have purchased the license, and you execute the tool, the distribution files that you need to distribute the applications are generated. These files are marked with an asterisk (*) below.

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

For all SQL Compare functions:

- SQLCompare.exe
- RedGate.SQLCompare.Distribution.dll*
- RedGate.SQLCompare.Distribution.mod*
- RedGate.SQL.Shared.sql
- RedGate.SQL.Shared.utils
- RedGate.SQLCompare.Engine.dll
- RedGate.SQLCompare.CommandLine.dll
- RedGate.SQLCompare.ASTParser.dll
- RedGate.SQLCompare.Rewriter.dll
- SQLCompare.exe.config
- RedGate.BackupReader.dll
- RedGate.BackupReader.sqbReader.dll

For reading backups only:

- System.Data.SQLite.dll

For reading encrypted backups:

- RedGate.BackupReader.CryptHelper.dll

For reading compressed backups:

- zlib.dll

Simple examples using the SQL Compare command line

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

For detailed examples of how to include specific tables, see:

- Example: selecting single tables for comparison
- Example: selecting tables with unrelated names

For a detailed list of switches used in the command line, see [Switches used in the SQL Compare command line](#).

For more general information on how to use a command line interface, you may also wish to refer to the topic [Working with command line interfaces](#).

Comparing and synchronizing 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 synchronize the databases by updating *WidgetProduction*:

```
sqlcompare /database1:WidgetStaging /database2:WidgetProduction  
/synchronize
```

To use a project that you have previously created using the graphical user interface and saved as "*widgets.scp*":

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

When you use a project, all objects that were selected for comparison when the project was saved are automatically included; you do not need to explicitly include them using the */include* switch. You can override the inclusions by specifying the */exclude* switch as required, for example if you do not want to compare any views:

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

Scheduling or automating a comparison or synchronization

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 7"  
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 synchronization of the two databases, updating the database *WidgetProduction*, you would create the script:

```
cd "C:\Program Files\Red Gate\SQL Compare 7"  
    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.

Exporting a schema

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.

To export *WidgetProduction* to a snapshot:

```
sqlcompare /database1:WidgetProduction  
    /makesnapshot:"C:\Snapshots\WP_1.snp"
```

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 synchronize 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 synchronization will be made writable. If you do not include the */force* switch and read-only files need to be modified, the synchronization 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 synchronization SQL script, for example for auditing purposes, and overwrite the file if it exists already:

```
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 using native SQL Server backups and the backup contains multiple backup sets, use the */backupsetname1* switch to specify the sets which make up the first backup, and use the */backupsetname2* switches to specify the sets which make up the second:

```
sqlcompare /backup1:"D:\MSSQL\BACKUP\WidgetDev.bak"  
          /backupsetname1:"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 */BackupPassword1* and */BackupPassword2* 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" /password1:Pa$$w0rd  
          /db2:WidgetLive
```

Example: Selecting single tables for comparison

This example illustrates how you 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 Examples using the command line.

Using the command line

To specify the table to include, you 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]*

Note that you use .NET standard regular expressions to define the */include* and */exclude* arguments. Therefore, 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`, all of the tables listed above are included, because they all contain 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 and SQL Server 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:\[dbo\]\ .\[Product\]
```

`/verbose` specifies that you want to display detailed information about differences between objects

Using XML

You can use XML as follows:

```
<?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 you 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 */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:\[Product\]^|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*

Note that you use .NET standard regular expressions to define the `/include` and `/exclude` arguments. Therefore, 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. (Note that 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 as follows:

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

Note that 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:

```
sqlcompare /argfile:XMLFileName.xml
```

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

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

Examples

Below are some examples of XML files that can be used with the SQLCompare tool. 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.

Note that 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 export the structure of a database to a scripts folder, an XML file containing some basic details about the database is generated automatically by SQL Compare. The file is called *SqICompareDatabaseInfo.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.

Command line syntax

Switches used in the command line

You can use the following switches with the SQL Compare command line:

/AllowIdenticalDatabases

Alias: */aid*

Lists all matching objects when compared databases are identical.

This option allows the comparison of two identical databases. By default, when the compared databases are identical, an error message is returned and no objects are listed. Use of this switch allows the comparison of identical databases. This is useful if you suspect two databases may be identical and want to verify this in detail:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProductio  
/AllowIdenticalDatabases
```

/argfile:<argfile>

Runs a file containing an XML argument specification:

```
sqlcompare /argfile:XMLFileName.xml
```

For more information see Using XML to specify command line arguments.

/backup1:<filename1>;<filename2>;...;<filenameN>

Alias: */bk1*

Allows one or more backup files to be specified in place of the first database:

```
sqlcompare /backup1:D:\BACKUPS\WidgetStaging.bak /db2:WidgetStaging
```

/backup2:<filename1>;<filename2>;...;<filenameN>

Alias: */bk2*

Allows one or more backup files to be specified in place of the second database:

```
sqlcompare /db1:WidgetStaging /backup2:D:\BACKUPS\WidgetStaging.bak
```

/backupPasswords1:<Password1>,<Password2>,...,<Password1N>

Alias: */bpsw1*

Specifies the password for the first backup:

```
sqlcompare /backup1:D:\BACKUPS\WidgetStaging.bak  
/BackupPasswords1:P@ssw0rd /db2:WidgetProduction
```

`/backupPasswords2:<Password1>,<Password2>,...,<Password1N>`

Alias: `/bpsw2`

Specifies the password for the second backup:

```
sqlcompare /db1:WidgetStaging
           /backup2:D:\BACKUPS\WidgetProduction.bak /BackupPassword2:P@ssw0rd
```

`/backupSetName1:<backupSet>`

Alias: `/bks1`

If you are using native SQL Server backups and the backup contains multiple backup sets, use the `/backupsetname1` switch to specify the sets which make up the first backup, and use the `/backupsetname2` switch to specify the sets which make up the second:

```
sqlcompare /Backup1:D:\BACKUP\WidgetDev.bak
           /BackupSetName1:"2008-09-23 Full Backup" /db2:WidgetDevelopment
```

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

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

```
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 `/backupperpassword1` and `/backupperpassword2` switches to specify the passwords; when there is more than one password, the passwords are separated using semicolons.

```
sqlcompare /Backup1:D:\BACKUPS\WidgetDev.sqb /BackupPassword1:Pa$$w0rd
           /db2:WidgetLive
```

`/backupSetName2:<backupSet>`

Alias: `/bks2`

Specifies which backup set to use for the second backup:

```
sqlcompare /db1:WidgetProduction /BackupSetName2:"2008-09-23 Full Backup"
```

`/database1:<database1>`

Alias: `/db1`

Specifies the first database to compare:

```
sqlcompare /database1:WidgetStaging /database2:WidgetProduction
```

`/database2:<database2>`

Alias: `/db2`

Specifies the second database to compare:

```
sqlcompare /database1:WidgetStaging /database2:WidgetProduction
```

`/exclude:<object type>:<regular expression>`

To specify the list of tables to exclude, you use the `/exclude` switch:

```
sqlcompare /db1:Customers1 /db2:Customers2 /exclude:table
```

`/exclude:table` specifies that you do not 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 type for exclusion use multiple `/exclude` switches. For example, to exclude only tables and views, enter:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
/exclude:table:WidgetReferences /exclude:view
```

For a more detailed explanation of the `/include` and `/exclude` switches, see [Selecting tables with unrelated names](#).

`/force`

Alias: `/f`

This forces the overwriting of any output files that already exist. If this switch is not used and a file of the same name already exists, the program will exit with the exit code indicating an IO error.

`/include:<object type>:<regular expression>`

Specifies which database objects or types are to be included in synchronization.

This switch is used to specify the list of tables to include. 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
```

For more detailed information on using the `/include` switch, see: [Selecting tables with unrelated names](#).

`/includeIdentical:<includeIdentical>`

Alias: `/inci`

Include identical objects in the reports. The values for this switch are: *True* or *False*. The default value is *False*.

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
/includeIdentical:True
```

`/makescripts:<folder>`

Alias: `/mkscr`

Creates a folder of scripts from the database specified by the switch: */database1*:

```
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:

```
sqlcompare /scr1:"C:\Scripts Folders\Widget dev scripts"  
  /scr2:"C:\Scripts Folders\Widget staging scripts" /synchronize
```

/makesnapshot:<filename>

Alias: */mksnap*

Creates a snapshot of the database specified by the switch: */database1*.

```
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.

/options:<option1>,<option2>,<option3>

Alias: */o*

Applies the project configuration options used during comparison or synchronization:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
  /options:Default,IgnoreWhiteSpace
```

For a detailed list of these options see: Options used in the command line.

/out:<fileName>

Redirects console output to the specified file:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction /out:C:\output file
```

/outputproject:<filename>

Alias: */outpr*

Writes the settings used for the comparison to the specified SQL Compare project file:

```
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 are not truncated, and that SQL script lines are not 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:<password1>

Alias: */p1*

The password for the first live database.

You must also provide a username. If you do not specify a username and password combination then integrated security is used:

```
sqlcompare /db1:WidgetStaging /username1:User1 /password1:P@ssw0rd  
          /db2:WidgetProduction /username2:User2 /password2:Pa$$w0rd
```

/password2:<password2>

Alias: */p2*

The password for the second live database.

You must also provide a username. If you do not specify a username and password combination then integrated security is used:

/project

Alias: */pr*

Uses a SQL Compare project (*.scp*) file for the comparison.

Option settings are not stored in the project, so if you wish to use anything other than the default options you must use the */options* switch to specify the options you want to use.

```
sqlcompare /project:WidgetProject.scp /options:AllowIdenticalDatabases
```

/report:<filename>

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
```

/reporttype:<reporttype>

Alias: */rt*

Defines the file format of the report produced by the */report* switch. The default setting is XML. The arguments for the */ReportType* switch are:

XML - Simple XML report
Simple - Simple HTML report
Interactive - Interactive report
Excel - Microsoft Excel spreadsheet

For example:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
  /report:"C:\reports\WidgetReport.html" /reporttype:Simple
```

/scriptencoding:<scriptencoding>

Alias: */senc*

Specifies the character encoding used when writing the SQL script file. The default is UTF8. The arguments for the */scriptencoding* switch are:

UTF8 - UTF-8 encoding, without preamble
UTF8WithPreamble - UTF-8 encoding, with 3-byte preamble
Unicode - UTF-16 encoding
ASCII - ASCII encoding

For example:

```
sqlcompare /db1:WidgetStaging /makescripts: D:\Scripts Folder  
  /scriptencoding:ASCII
```

scriptfile:<scriptfile>

Alias: */sf*

This 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:

```
sqlcompare /db1:WidgetStaging /db2:WidgetProduction  
  /scriptfile: "C:\Scripts Folder\WidgetSyncScript.sql"
```

/scripts1:<folder>

Alias: */scr1*

Allows a folder of scripts to be specified in place of the first database:

```
sqlcompare /scripts1:"C:\Scripts Folder\WidgetStagingScript"  
  /db2:WidgetProduction
```

/scripts2:<folder>

Alias: */scr2*

Allows a folder of scripts to be specified in place of the second database:

```
sqlcompare /db1:WidgetStaging
           /scripts2:"C:\Scripts Folder\WidgetStagingScript"
```

/server1:<server1>

Alias: /s1

This specifies the server on which the first (/db1:) database (or script, snapshot, or backup) is located. If an explicit path is not specified, it defaults to *Local*.

```
sqlcompare /Server1:Widget_Server\SQL2008 /db1:WidgetStaging
           /db2:WidgetProduction
```

/server2:<server2>

Alias: /s2

This specifies the server on which the second (/db2:) database (or script, snapshot, or backup) is located. If an explicit path is not specified, it defaults to *Local*.

```
sqlcompare /db1:WidgetStaging /Server2:Widget_Server\SQL2008
           /db2:WidgetProduction
```

/snapshot1:<filename>

Alias: /sn1

Specifies a snapshot to be used in place of the first database (/db1):

```
sqlcompare /Snapshot1:"C:\Snapshots\WidgetStagingSnapshot.snp"
           /db2:WidgetProduction
```

/snapshot2:<filename>

Alias: /sn2

Specifies a snapshot to be used in place of the second database (/db2):

```
sqlcompare /db1:WidgetStaging
           /snapshot2:"C:\Snapshots\WidgetProductionSnapshot.snp"
```

/synchronize or /scynchronise

Alias: /sync

Synchronizes the databases after comparison.

Changes are migrated from <database1> to <database2>:

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

`/username1:<username1>`

Alias: `/u1`

The username for the first database.

If no username is specified then integrated security is used.

```
sqlcompare /db1:WidgetStaging /username1:User1 /password1:P@ssw0rd  
          /db2:WidgetProduction /userName2:User2 /password2:Pa$$w0rd
```

`/username2:<username2>`

Alias: `/u2`

The username for the second database.

If no username is specified then integrated security is used:

```
sqlcompare /db1:WidgetStaging /username1:User1 /password1:P@ssw0rd  
          /db2:WidgetProduction /userName2:User2 /password2:Pa$$w0rd
```

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 do not explicitly set any options, the defaults are used. See *Defaults* below.

Defaults

If you do not specify any options, the following default options apply:

- DecryptPost2KEncryptedObjects
- IgnoreFillFactor
- IgnoreWhiteSpace
- IncludeDependencies
- IgnoreFileGroups
- IgnoreUserProperties
- IgnoreWithElementOrder

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 do not specify the *default* argument, only the options you do specify apply.

To specify no options, use the *none* argument.

Further options are detailed below.

AddWithEncryption

Alias: *we*

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

Note that if you use ADD ENCRYPTION on a SQL Server 2005 database, SQL Compare will not subsequently be able to compare, or synchronize the encrypted objects.

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

CaseSensitiveObjectDefinition

Alias: *cs*

For databases with case-sensitive collation, enables objects with case-sensitive names to be compared and synchronized. 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.

Note that you should take care 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 without this option set, SQL Compare may produce unexpected errors.

ConsiderNextFilegroupInPartitionSchemes

Alias: *cfgps*

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 synchronization if the partition scheme is extended. The next filegroup does not affect the way in which data is stored

DecryptPost2KEncryptedObjects

Alias: *dp2k*

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 2005 and SQL Server 2008 databases which were created using the WITH ENCRYPTION option. Note that when comparing large databases with few encrypted objects, selecting this option may result in slower performance.

When this option is not selected, text objects in SQL Server 2005 and SQL Server 2008 databases are shown as different, and cannot be synchronized.

DisableAndReenableDdlTriggers

Alias: *drd*

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

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

DoNotOutputCommentHeader

Alias: *nc*

When this option is set comments and comment headers are not included in the output synchronization script.

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 synchronization. Data will be preserved.

IgnoreBindings

Alias: *ib*

Ignores bindings on columns and user-defined types when comparing and synchronizing (e.g. `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. Consequently, SQL Compare cannot compare all of the properties for a symmetric key.

If certificates, symmetric keys, and asymmetric keys are selected for synchronization, only the permissions are synchronized.

IgnoreChecks

Alias: *ich*

Ignores check constraints when comparing and synchronizing databases.

IgnoreCollations

Alias: *ic*

Ignores collation orders on character datatype columns when comparing and synchronizing databases.

IgnoreComments

Alias: *icm*

Ignores comments when comparing views, stored procedures and so on. Comments will still appear in the synchronization scripts.

IgnoreConstraintNames

Alias: *icn*

Ignores the names of indexes, foreign keys, primary keys, and default, unique, and check constraints when comparing databases. Note that the names will not be ignored when the databases are synchronized.

IgnoreExtendedProperties

Alias: *ie*

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

IgnoreFileGroups

Alias: *ifg*

Ignores *filegroup* clauses, partition schemes, and partition functions on tables and keys when comparing and synchronizing databases. Partition schemes and partition functions are not displayed in the comparison results.

IgnoreFillFactor

Alias: *if*

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

IgnoreFullTextIndexing

Alias: *ift*

Ignores full-text catalogs and full-text indexes when comparing and synchronizing databases.

IgnoreIdentityProperties

Alias: *iip*

Ignores the identity property on columns when comparing databases. Note that the identity property will not be ignored when databases are synchronized.

IgnoreIdentitySeedAndIncrement

Alias: *isi*

For identity properties, ignores only the identity seed and increment values when comparing databases. Note that they will not be ignored when the databases are synchronized.

IgnoreIndexes

Alias: *ii*

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

IgnoreIndexLockProperties

Alias: *iilp*

Ignores the lock properties of indexes.

IgnoreInsteadOfTriggers

Alias: *iit*

Ignores INSTEAD OF DML triggers when comparing and synchronizing databases.

IgnoreKeys

Alias: *ik*

Ignores foreign keys when comparing and synchronizing databases.

IgnoreNotForReplication

Alias: *infr*

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

If you select this option, the NOT FOR REPLICATION statement will not 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 note that 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 note that this will apply to constraints in all objects.

IgnorePermissions

Alias: *ip*

Ignores permissions on objects when comparing and synchronizing databases.

IgnoreQueueEventNotifications

Alias: *iqen*

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

Ignores the event notification on queues when comparing and synchronizing 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. Note that these statements will not be ignored when the databases are synchronized.

IgnoreReplicationTriggers

Alias: *irpt*

Ignores replication triggers when comparing and synchronizing databases.

IgnoreStatistics

Alias: *ist*

Ignores statistics when comparing and synchronizing databases.

IgnoreTriggerOrder

Alias: *ito*

This option is used for SQL Server 2000 and later databases.

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 synchronizing databases. Note that the DDL trigger order is not affected.

IgnoreTriggers

Alias: *it*

Ignores DML triggers when comparing and synchronizing databases.

IgnoreUserProperties

Alias: *iup*

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

If this option is not 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 synchronization, SQL Compare synchronizes the properties where possible.

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

IgnoreUsers

Alias: *iu*

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

IgnoreWhiteSpace

Alias: *iw*

Ignores white space (newlines, tabs, spaces, and so on) when comparing databases. Note that white space will not be ignored when the databases are synchronized.

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), select this option to ignore the order of the WITH elements when comparing and synchronizing databases.

IgnoreWithNocheck

Alias: *iwn*

Ignores the WITH NOCHECK argument on foreign keys and check constraints. Ignores the 'not trusted' flag on foreign keys and check constraints.

Note that foreign keys or constraints that are *disabled*, are not ignored.

IncludeDependencies

Alias: *incd*

Includes dependent objects when comparing and synchronizing databases. For example, if a view depends on a table then the table will be synchronized when synchronizing the view.

none

Alias: *n*

To specify no options, use the *none* argument.

NoSQLPlumbing

Alias: *np*

Do not include plumbing for transactional synchronization scripts. Removes transactions from the synchronization SQL scripts to produce SQL code that is more readable.

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

UseClrUdtToStringForClrMigration

Alias: *uclr*

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

If CLR objects are to be synchronized, 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 synchronization.

This option affects the synchronization only.

Exit codes used in the SQL Compare 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 detailed below:

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* cannot 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.

63 - Databases identical

The databases being compared are identical or no objects have been included.

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.

73 - Failed to create report

The report was not created.

74 - I/O error

For example, 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 cannot be completed because the user does not have the necessary permission.

Getting started with SQL Changeset



SQL Changeset is a free tool that enables you to integrate your 'check out/edit/check in' (VSS style) source control system with SQL Compare Professional Edition.

When SQL Changeset is running, SQL Compare can retrieve the latest version of a database from your source control system immediately before comparing it with another database, or performing a synchronization, to ensure that you have the most up-to-date information in your comparison results.

SQL Source Control

Red Gate now offers SQL Source Control (http://www.red-gate.com/products/SQL_Source_Control/index.htm), a tool to integrate database source control with SQL Server Management Studio.

Note that SQL Changeset will be deprecated with a future release of SQL Compare.

Supported programs

SQL Changeset is installed with SQL Compare 6.2 Professional Edition or later.

The following source control systems are supported:

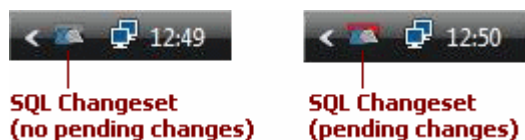
- Microsoft® Visual SourceSafe® versions 6.0d and 2005
- Microsoft® Visual Studio® 2005 Team Foundation Server SP1

SQL Changeset may also work with other source control systems, but they are not currently supported.

For more information, see [Source control systems supported by SQL Changeset](#)

Using SQL Changeset

SQL Changeset is a notification area program that you can launch from your Start menu, or from within SQL Compare. When SQL Changeset is running, the SQL Changeset icon is displayed in the notification area, at the far right of the taskbar.



You are recommended to register your working folders with SQL Changeset before you start SQL Compare. To find out how to do this, see [Registering working folders](#).

You can use SQL Changeset to view the pending changeset for files contained in your registered working folders. This includes all changes, not only those made by SQL Compare. For details, see [Committing Changes](#) (page 111).

For examples of how you can use SQL Changeset with SQL Compare, see:

- [Getting the latest database \(page 96\)](#)
- [Saving modifications to source \(page 104\)](#)

Getting the latest database

This example demonstrates how to get the latest version of a database that has been stored as a set of script files (a *scripts folder*) in a source control system. You do this by using SQL Compare with SQL Changeset to synchronize the schema of a local database with the scripts folder.

If you are not familiar with using SQL Compare, you are recommended to follow the worked examples provided in the SQL Compare help before you read this.

In the example, the Super Gizmo Company has a SQL Server database running on a live server. This database contains a number of tables, views, stored procedures, and other database objects. The Super Gizmo Company's development team has been given the task of making a number of changes to the structure of the database. The allocated developer needs to ensure that his local development version of the database is identical to the latest version of the database in source control.

This example uses two databases:

- GizmoProduction is the production database.
This database has been exported to a scripts folder, and the scripts folder is stored in the source control system.
- GizmoDevelopment is the developer's local version of the production database; this is not up-to-date initially.


If you would like to follow the example on your own system, you need to set up two databases; one that has been exported to a scripts folder and stored in your source control system, and one that is live on your SQL Server.

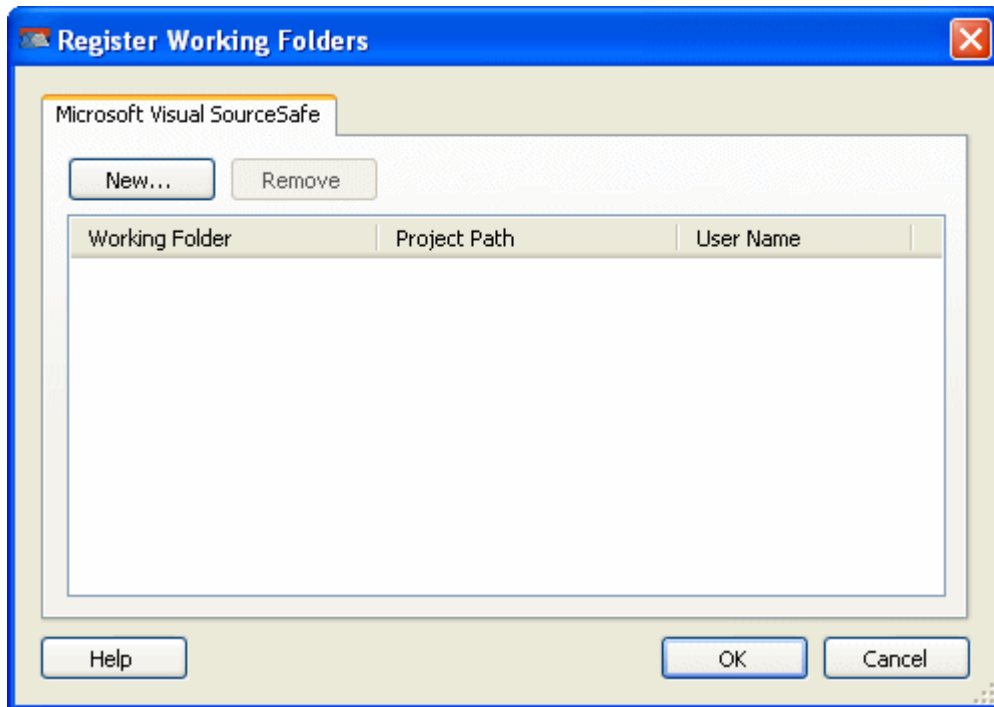
Registering the working folder

The first time you use a particular working folder with SQL Changeset, you must register the working folder with SQL Changeset. Once it has been registered, SQL Changeset remembers it and you can omit this step in future.

1. From the **Start** menu, select **SQL Changeset**.

If you have not yet registered any working folders, the **Register Working Folders** dialog box is displayed.

If you have already registered working folders in SQL Changeset, your source control system may prompt you to enter your logon details (the logon dialog box for your source control system will be displayed; you may need to use Alt+Esc to switch to the dialog box). The Pending Changeset dialog box is then displayed. Close this dialog box, and in the notification area, right-click  **SQL Changeset**, and then click **Register Working Folders**.

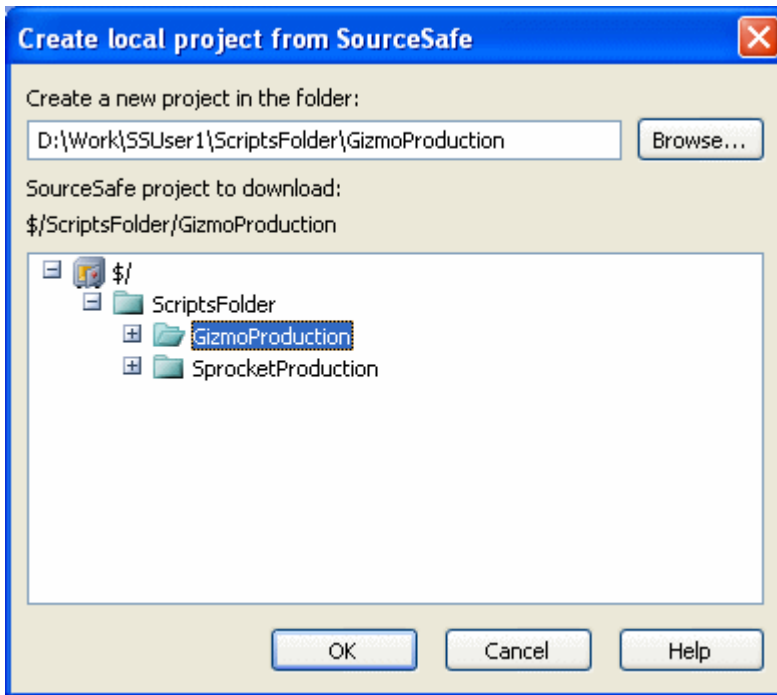


SQL Changeset displays a tab for each source control system client it detects. Any registered working folders are listed.

2. Select the tab for the source control system you want to use, and then click **New**.

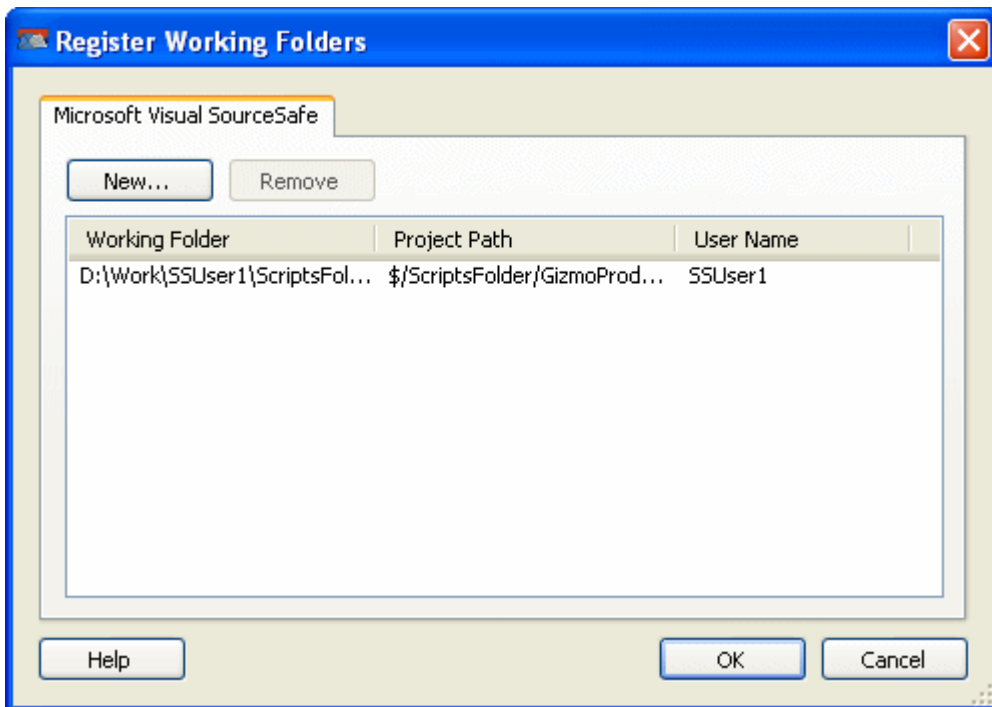
If prompted, enter the authentication details as required by your source control system.

The source control system displays a dialog box for you to set up the working folder. For example, Microsoft® Visual SourceSafe® displays the **Create local project from SourceSafe** dialog box.



3. Specify the working folder, and then click **OK**.

In this example, the production database scripts folder is *ScriptsFolder\GizmoProduction*. The registered working folder is listed in the **Register Working Folders** dialog box.



4. Click **OK**.

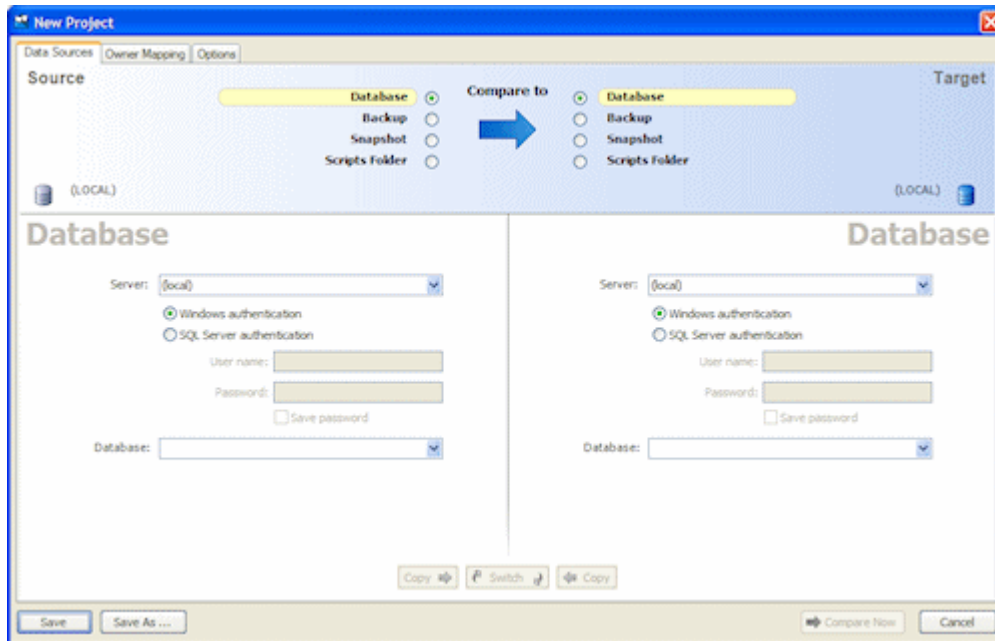
The working folder you selected is registered.

Comparing the scripts folder with the developer's local version

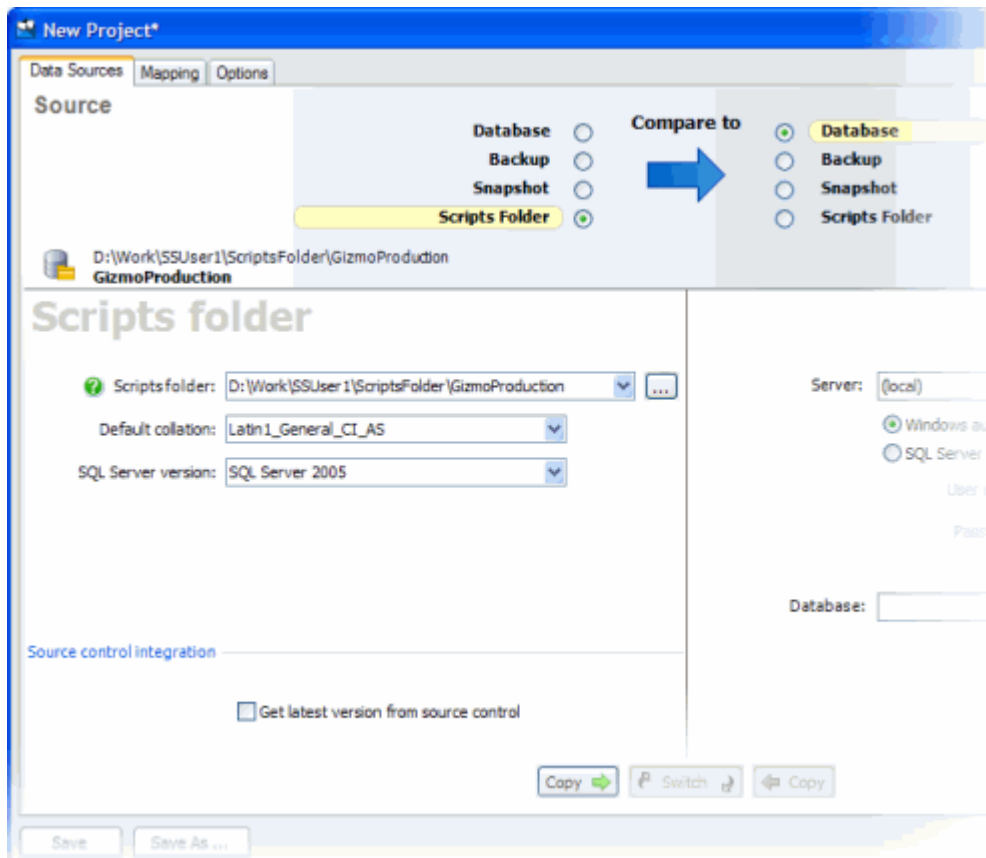
To compare the schema saved as a scripts folder in source control with the developer's database, you set up a comparison project:

1. In SQL Compare, on the **Projects** dialog box, click  **New**.

The **Project Configuration** dialog box is displayed.



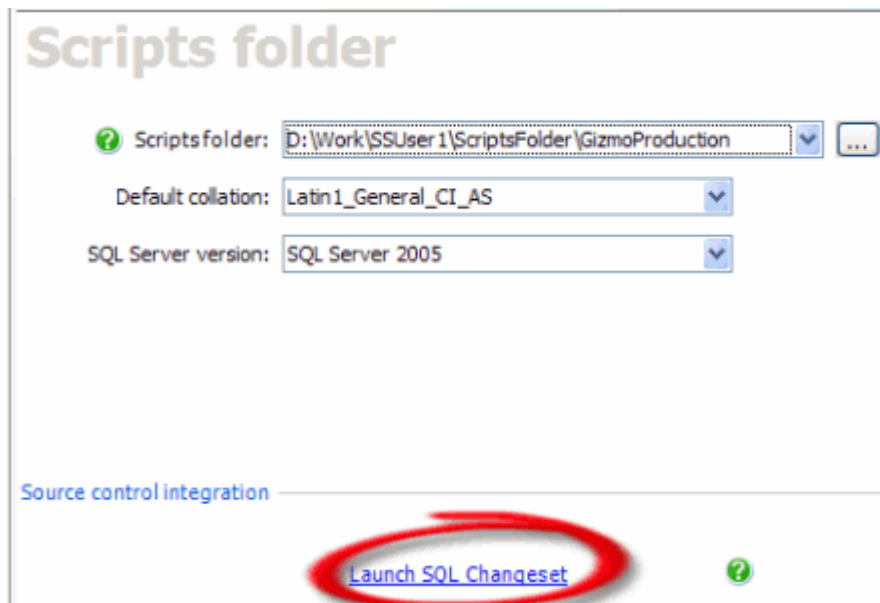
Set the Source on the left to *Scripts folder*, and browse to the local working folder that you registered in SQL Changeset.



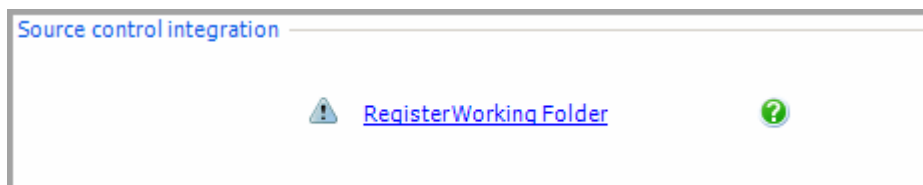
2. Select the **Get latest version from source control** check box.

This ensures that you will compare the latest version of the production database. The files in the local working folder will be overwritten when you click **Compare Now**. (Note that SQL Compare will get the latest files again if you re-run this comparison project by clicking **Compare Now** on the SQL Compare **Comparison Projects** dialog box. However, if you click **Refresh Comparison** on the main results page, SQL Compare will *not* get the latest files.)

If SQL Changeset is not already running, the **Get latest version from source control** check box is not available; click the **Launch SQL Changeset** link to start SQL Changeset.

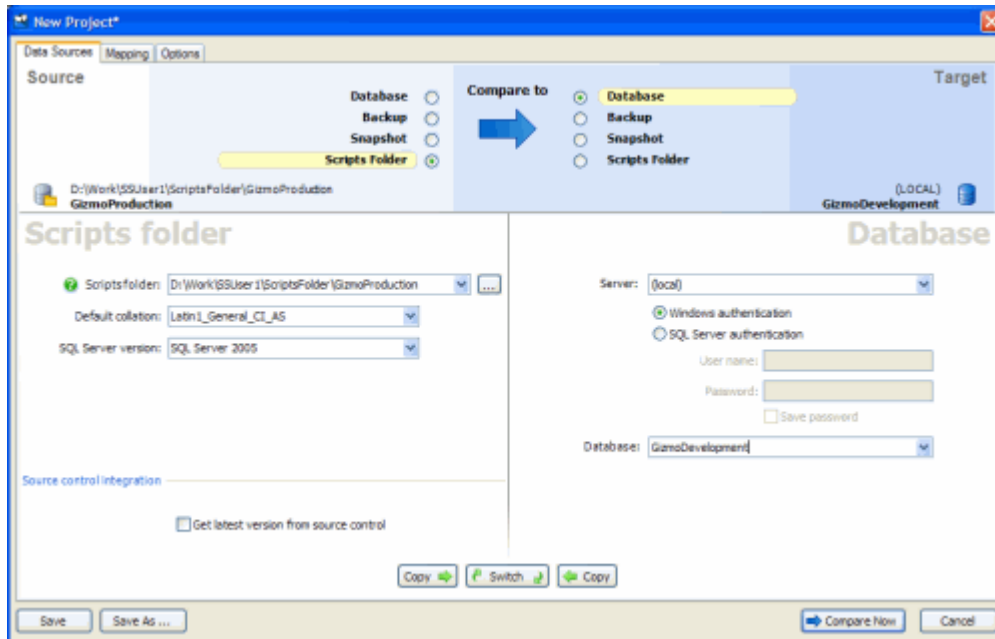


The **Get latest version from source control** check box is also unavailable if SQL Changeset is running, but you have not yet registered the working folder in SQL Changeset; you can open the SQL Changeset Register Working Folders dialog box by clicking the link on the **Project Configuration** dialog box:



3. On the right of the **Project Configuration** dialog box, under Target, set the **Data source type** to **Database**, and select the SQL Server and the developer's local database.

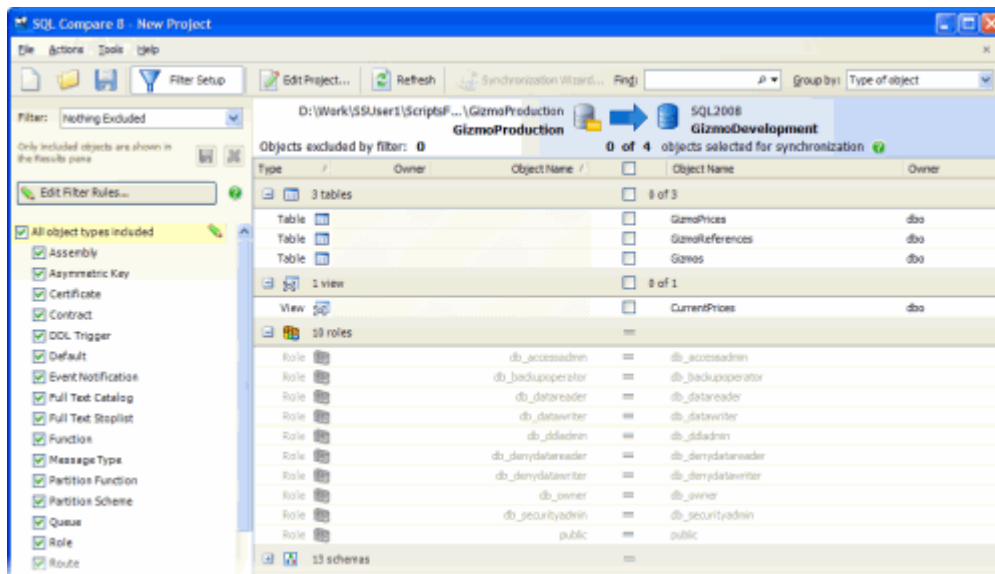
In this example, the GizmoDevelopment database on the local server is the database that will be updated.



4. Click **Compare Now**.


A message dialog box is displayed. If you selected the **Close dialog box on completion** check box previously, SQL Compare closes this message dialog box automatically.

SQL Compare displays the differences between the latest version of the database schema in source control (GizmoProduction), and the development database (GizmoDevelopment).



Updating the development database

You want to update the development database schema. To do this:

1. Ensure all the objects are selected for synchronization, and then click  **Synchronization Wizard**.
2. On the **Choose Synchronization Method** page, select **Synchronize Using SQL Compare**.
3. Click **Next** to view the **Dependencies** page (there are no dependencies in this example), and then click **Next** again to view the **Review Script** page.

For more information about these pages, see Using the Synchronization wizard.

4. Click **Synchronize Now** to update the development version of the database.
SQL Compare synchronizes the databases, then re-compares them. A message dialog box shows the progress of the comparison.
5. Click **OK** on the progress dialog box, if necessary.

The developer's local database is now identical to the production database in source control, and the developer can proceed with the modifications.

To see how the developer updates the source control version when the modifications have been made, see Saving modifications to source control (page 104).

Saving modifications to source control

This example demonstrates how to use SQL Changeset to update a database that has been stored as a set of script files (a *scripts folder*) in a source control system by synchronizing with the schema of a local version of the database.

Note that the local copy of the scripts folder must be the latest version. If the files in the source control system have been updated since you retrieved them (for example, by another developer), you must merge the differences into the local live database and your local copy of the files before you proceed with the synchronization.

If you are not familiar with using SQL Compare, you are recommended to follow the worked examples provided in the SQL Compare help before you read this.

In the example, the Super Gizmo Company's developer has made a number of changes to the structure of the GizmoProduction database on a local copy of the database. The developer has tested the changes locally on the GizmoStaging database, and now wants to update the source control version of GizmoProduction with the changes.

This example uses two databases:

- GizmoProduction is the production database
This database has been exported to a scripts folder. The scripts folder is under source control and registered in SQL Changeset as a working folder.
- GizmoStaging is the developer's local version of the production database that contains the modifications.

If you would like to follow the example on your own system, you need to set up two databases; one that has been exported to a scripts folder, stored in your source control system, and registered in SQL Changeset, and one that is live on your SQL Server.

Comparing the developer's local version with the scripts folder

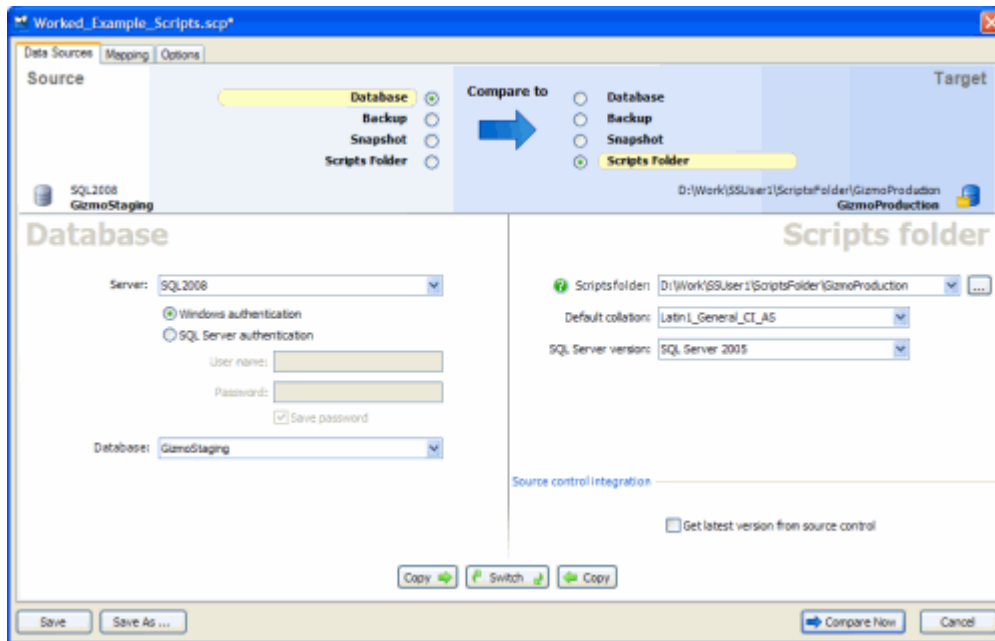
To compare the developer's modified database with the latest version of the production database in the source control system, set up a comparison project:

In SQL Compare, on the **Projects** dialog box, click  **New**.

1. The **Project Configuration** dialog box is displayed.
2. On the left of the **Project Configuration** dialog box, set the Source to **Database**, and select the SQL Server and the database.

In this example, the GizmoStaging database on the local server is selected.

- Set the Target on the right to *Scripts folder*, and browse to the local working folder for the scripts folder.



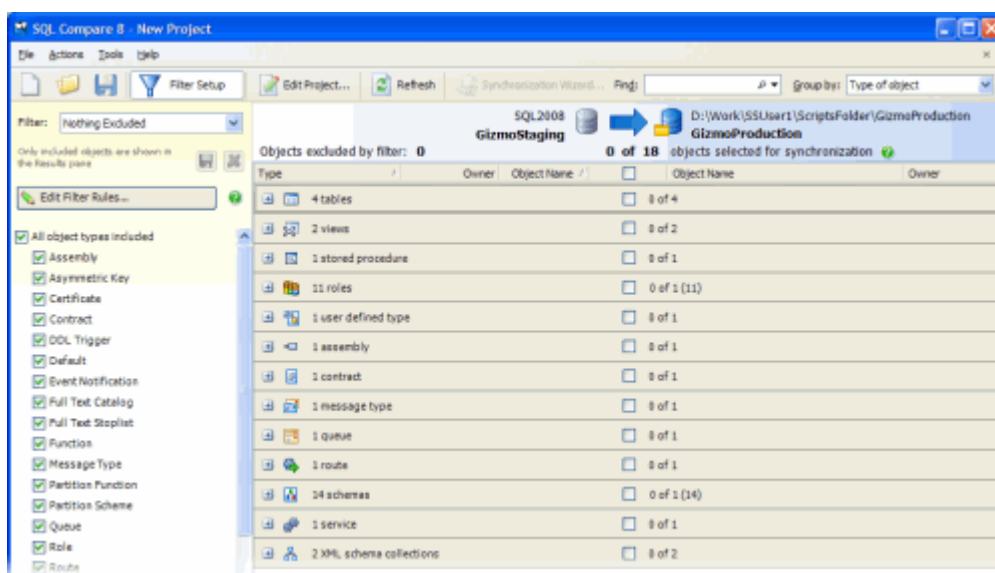
- Clear the **Get latest version from source control** check box.

The latest version was retrieved from source control before the modifications were made; see Getting the latest database (page 96) for details.

- Click **Compare Now**.


A message dialog box is displayed. If you selected the **Close dialog box on completion** check box previously, SQL Compare closes this message dialog box automatically.

SQL Compare displays the differences between the local version of the production database and the staging database.

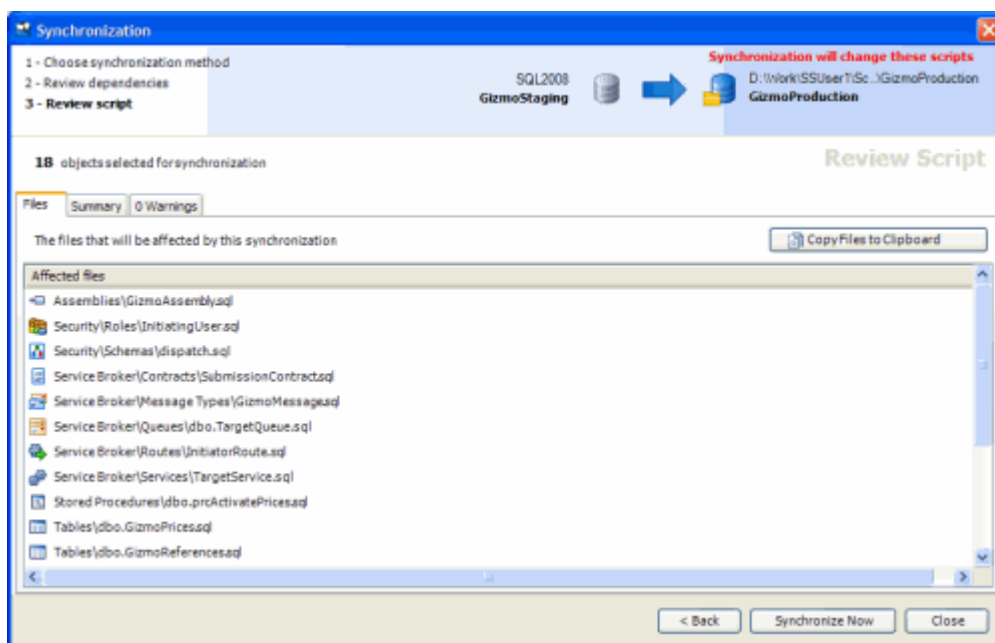


Updating the production database scripts folder

You want to update the scripts folder for the production database. To do this:

1. Ensure all the objects are selected for synchronization, and then click  **Synchronization Wizard**.
2. On the **Choose Synchronization Method** page, ensure that **Synchronize using SQL Compare** is selected.
3. Click **Next** to view the **Dependencies** page; there are no dependencies in this example.
4. Click **Next** again to view the **Review Script** page.

The **Files** tab lists the files that will be created or modified.



If any of the files to be modified are not the latest version, SQL Compare displays a warning in the **Warnings** tab. You must merge the differences into the live development database and your local copy of the files before you proceed with the synchronization.

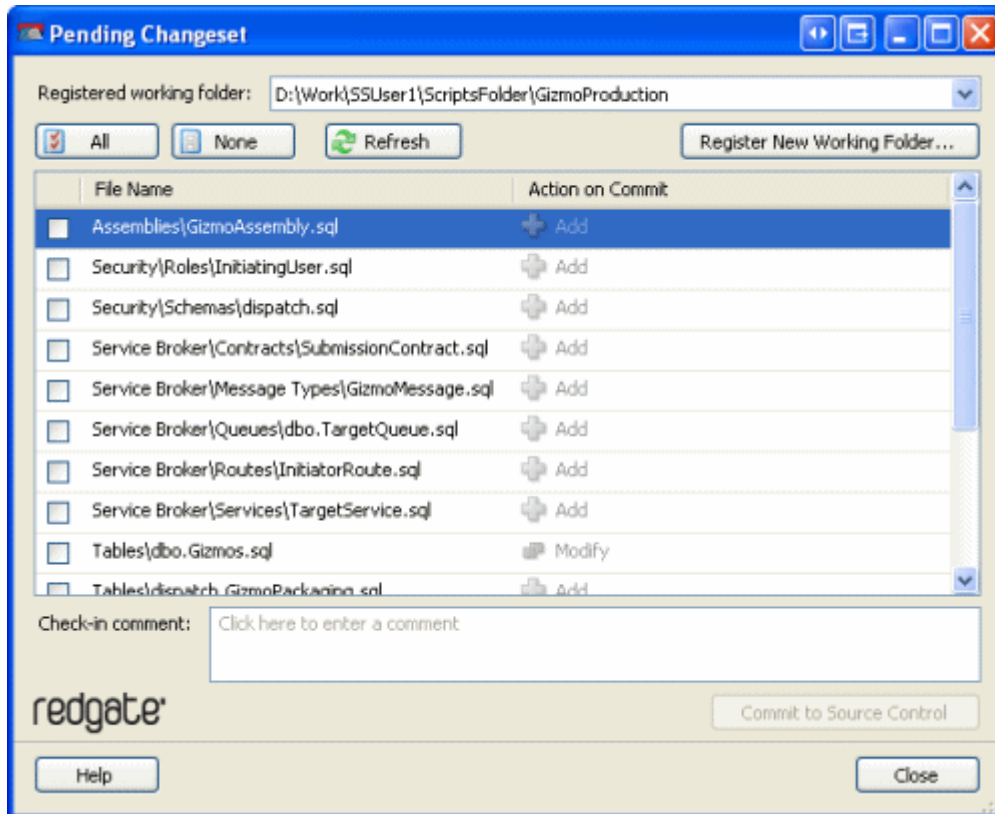
5. Click **Synchronize Now** to update the scripts folder.
SQL Compare synchronizes the databases, then re-compares them. A message dialog box shows the progress of the comparison.
6. If necessary click **OK** on the Progress dialog box.

Committing the changes

SQL Compare has checked out and modified some of the files in the scripts folder, and created some new files. The developer can now check the changes in to the source

control system using SQL Changeset. SQL Compare does not need to be running for you to commit the changes.

1. In the notification area, double-click  **SQL Changeset**, or right-click  **SQL Changeset** and then click **Open Pending Changeset**.



2. Select the changes that you want to commit.

In this example, click **All**.

3. In the **Comment** box, type a comment for these changes.
4. Click **Commit to Source Control**.

The changes are committed, and the list is refreshed. The files are checked in to the source control system. If there are any issues with the check in, for example, if any of the files have been amended in the source control system and require a merge, your source control system will prompt you in the usual way.

5. Click **Close** to close the dialog box.


The GizmoProduction scripts folder in the source control system has now been updated with the developer's modifications.

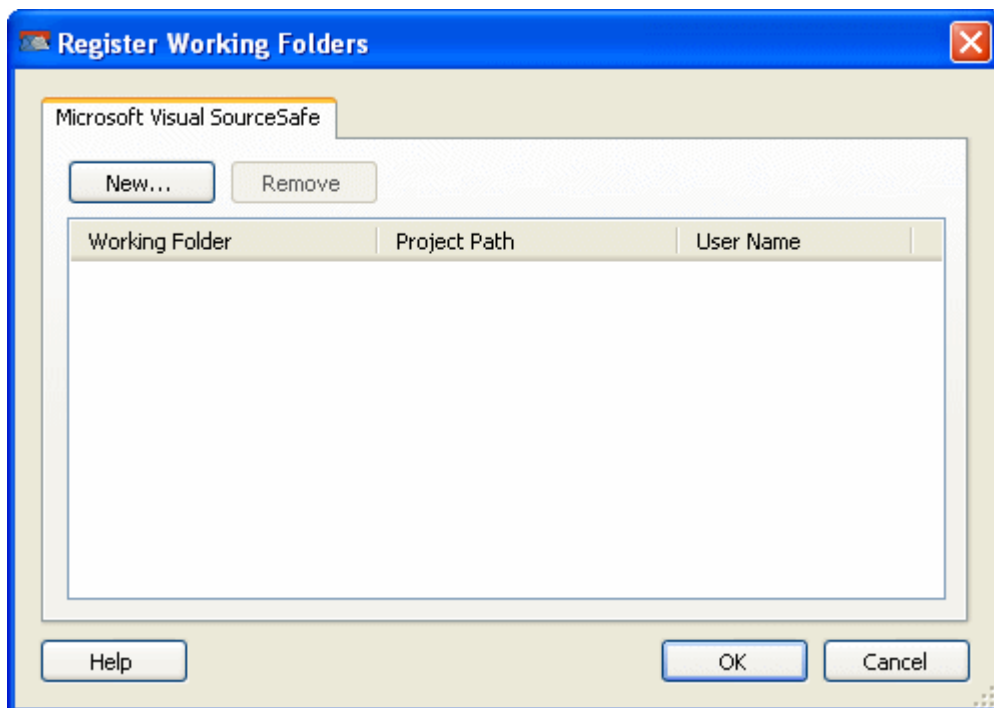
Registering working folders

To use SQL Changeset to check in and check out files that are under source control, you must register the working folder with SQL Changeset. When you have registered a working folder, SQL Changeset remembers it until you remove the registration.

1. If SQL Changeset is not already running, from the Start menu, select SQL Changeset.

If you have not yet registered any working folders, the **Register Working Folders** dialog box is displayed.

If you have already registered working folders in SQL Changeset, your source control system may prompt you to enter your logon details. The **Pending Changeset** dialog box is then displayed. Close this dialog box, and in the notification area, click  **SQL Changeset**, and then click **Register Working Folders**.

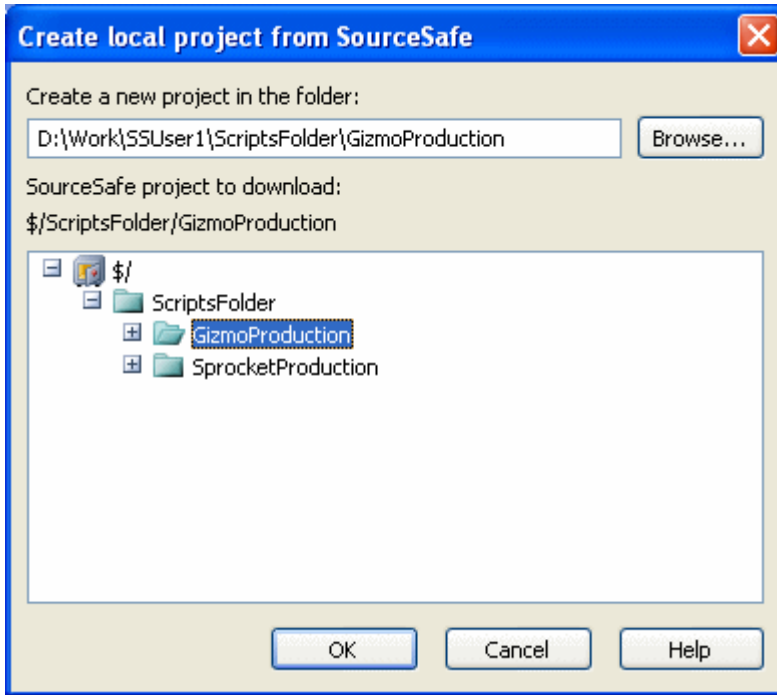


A tab is displayed for each source control system client detected on your computer. However, note that not all source control systems are supported (see Using SQL Changeset (page 94)). SQL Changeset displays a warning on the tab if the source control system is not supported.

2. Select the tab for the source control system, and click **New**.

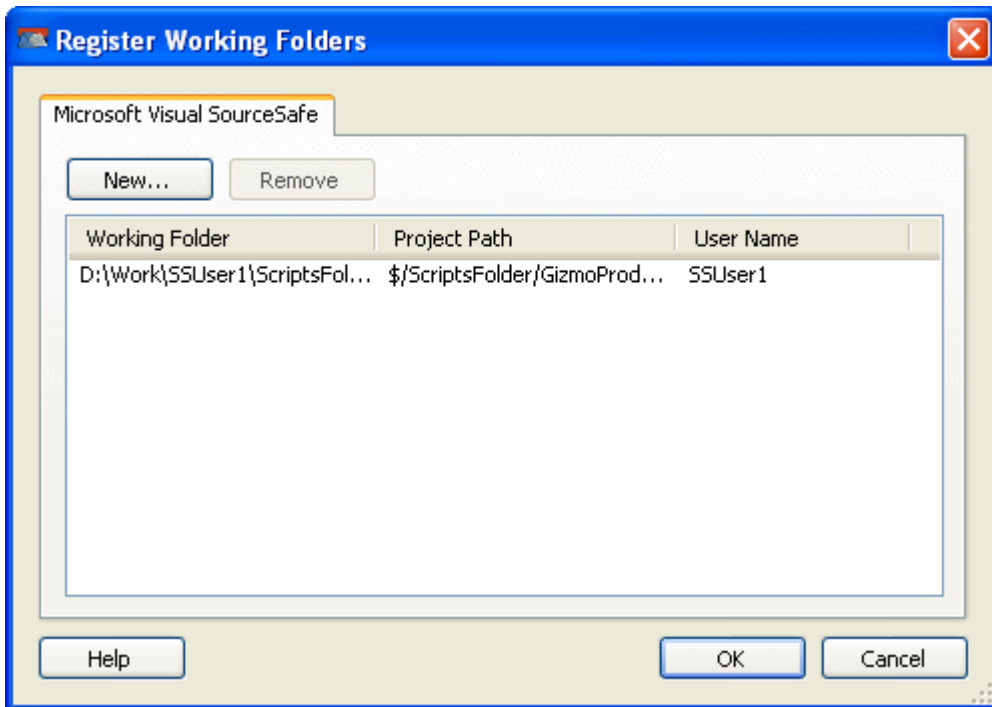
Your source control system may prompt you for authentication details. You should use the same user name for all working folder registrations.

The source control system that you selected displays a dialog box for you to select the working folder. For example, Microsoft® Visual SourceSafe® displays the **Create local project from SourceSafe** dialog box.



3. Specify the working folder, and then click **OK**.

The details of the project are displayed in the **Register Working Folders** dialog box.



You can add more working folders as required.

4. Click **OK**.



The working folders are registered. If there are any changes pending in a registered working folder, the **Pending Changeset** dialog box is displayed. For details, see [Committing changes \(page 111\)](#).

To remove the registration for a working folder, click the working folder in the list, and then click **Remove**.

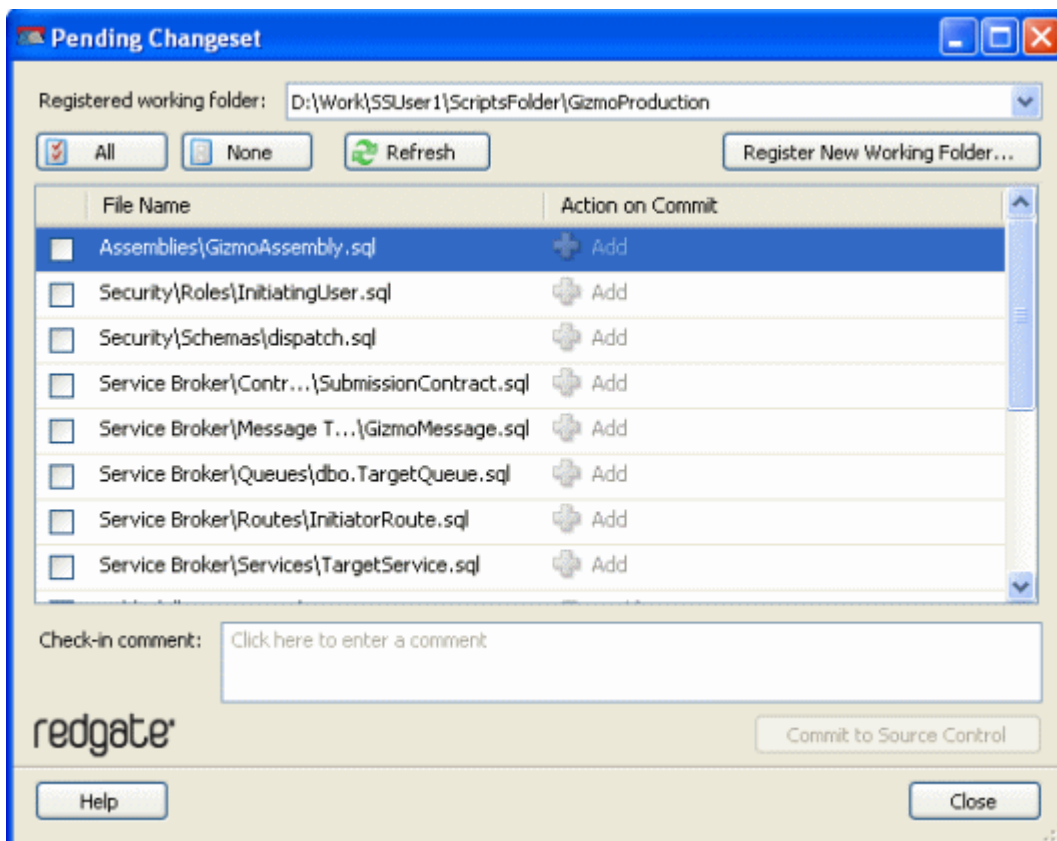
Committing changes

You can use SQL Changeset to view the pending changeset for files contained in working folders that you have registered with SQL Changeset. This includes all changes, not only those made by SQL Compare.

The color of the SQL Changeset icon in the notification area indicates whether there are changes to be committed:

-  indicates pending changes
-  indicates no changes are pending

If you have registered working folders, when you start SQL Changeset you are prompted to log on to your source control system, and then the **Pending Changeset** dialog box is displayed. If this dialog box is not displayed (for example because you have closed it), double-click the icon, or right-click the icon and then click **Open Pending Changeset**.



(If you are using Team Foundation Server, the logon dialog box may be displayed behind the **Pending Changeset** dialog box, and the **Pending Changeset** dialog box will appear to be waiting for information. Press Alt+Esc to display the logon dialog box.)



You can select a different registered working folder to view from the drop-down list.

To view the local version of the file, right-click the file and then click **View Local Version**; to see the changes that have been made to a file, right-click the file, and click **Show Differences**.

Select the check boxes next to the files for which you want to commit changes, type a comment, and then click **Commit to Source Control**.

To select multiple files, use Ctrl and Shift in the usual way to highlight the files, then right-click and click **Select Items**.

Note that when you commit the changes, your source control system may display message boxes, such as error messages. These messages may be displayed behind any other windows that you have open, and so may not be visible. Unfortunately, SQL Changeset is unable to access or process these errors. To proceed, you must use Alt+Esc to access the message box, and then clear it, for example by clicking **OK**. Refer to your source control system documentation for more information on specific errors or messages.

If the changes are committed successfully,  is displayed. If there is a problem committing the changes  is displayed; move your mouse pointer over the icon to see a tooltip containing the error message from your source control system.

Troubleshooting SQL Changeset

This topic provides information about some issues that may arise when you are using SQL Changeset.

Errors

When you use SQL Changeset, you may see errors that are raised by your source control system. Unfortunately, due to the interface provided by the generic source control API, SQL Changeset is unable to access or process these errors. To proceed, you must clear the error message, for example by clicking **OK**. Refer to your source control system documentation for more information on specific errors or messages.

Note that the error messages are sometimes displayed behind the SQL Changeset (or SQL Compare) windows, which may give the appearance that SQL Changeset is not responding. Use Alt+Esc to access the error message and clear it.

Failed synchronization

If SQL Compare cannot check out all the files when synchronizing, the synchronization process is stopped and a message is displayed. Note that if SQL Compare has already checked out some files before the synchronization is stopped, those files remain checked out; undo the check out for the files in your source control system.

SQL Changeset is not responding

Your source control system may display a message box (such as an error notification) or an authentication dialog box behind your current windows; this gives the appearance that SQL Changeset is not responding. Unfortunately, due to the interface provided by the generic source control API, SQL Changeset is unable to access or process these message boxes.

To proceed, you must use Alt+Esc to access the message box, and then clear it, for example by clicking **OK**. Refer to your source control system documentation for more information on specific errors or messages.

Visual Studio 2005 Team Foundation Server not detected

When you display the Register Working Folders dialog box, SQL Changeset automatically detects any SCCI-compliant source control applications installed on your computer. If you have installed Microsoft® Visual Studio® 2005 Team Foundation Server SP1, but SQL Changeset has not detected it, ensure that you have installed the Visual Studio Team Foundation Server MSSCCI Provider.

Deleted files in source control system

If the files in a working folder that you have registered with SQL Changeset are deleted in the source control system by another user, SQL Changeset lists the missing files in the **Pending Changeset** dialog box with **Add** as the action on commit. This is because SQL Changeset assumes that the local files are new because they do not exist in the source control system.

If you are using Visual Studio® 2005 Team Foundation Server and you 'get latest', the redundant files are deleted. When you refresh the pending changeset list, the files no longer appear in the list.

If you are using Visual SourceSafe® and you 'get latest', the local files are not deleted. If you do not want to keep the files, you must delete them manually.

Working folders registered for different users

You are advised to register all working folders using only one user name. If you register working folders using different user names, this may cause error messages or failure when you use a working folder that is registered under a user name that is different from the one you are currently using.

If you have already registered working folders using different user names, remove the registrations for the different user names, and re-register the working folders.

Unicode support

Due to limitations of SCCI implementation, SQL Changeset does not support Unicode.

Getting Technical Support

Further support resources such as forums, FAQs, and knowledge bases, are available in our support site.

The support team may ask you to send a log of SQL Changeset's activities; this is located in:

C:\Documents and Settings\\Local Settings\Application Data\Red Gate\SQL Changeset\Log <datetime>.txt

In Vista it is located in:

C:\Users\\AppData\Local\Red Gate\SQL Changeset\Log <datetime>.txt

Acknowledgements

Trademarks and registered trademarks

Red Gate is a registered trademark of Red Gate Software Ltd registered in the U.S. Patent and Trademark Office.

.NET Reflector and SQL Compare are registered trademarks of Red Gate Software Ltd registered in the U.S. Patent and Trademark Office.

ANTS Performance Profiler, ANTS Memory Profiler, .NET Reflector Pro, Exception Hunter, Schema Compare for Oracle, SQL Backup, SQL Data Compare, SQL Comparison SDK, SQL Dependency Tracker, SQL Doc, SQL Log Rescue, SQL Multi Script, SQL Packager, SQL Prompt, SQL Refactor, SQL Response, SQL Toolbelt, and Exchange Server Archiver are trademarks of Red Gate Software Ltd.

Microsoft, Windows, Windows 98, Windows NT, Windows 2000, Windows 2003, Windows XP, Windows Vista, Windows 7, Visual Studio, and other Microsoft products referenced herein are either registered trademarks or trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

InstallShield is a registered trademark and service mark of InstallShield Software Corporation.

Copyright information

All Red Gate applications are © Red Gate Software Ltd 1999 - 2010

SQL Backup, SQL Compare, SQL Data Compare, SQL Packager, and SQL Prompt contain software that is Copyright © 1995 - 2005 Jean-loup Gailly and Mark Adler.

SQL Doc includes software developed by Aspose (<http://www.Aspose.com>).

SQL Backup contains software that is Copyright © 2003 - 2008 Terence Parr. Refer to the ACKNOWLEDGEMENTS.txt file in your SQL Backup installation directory for the full license text.