SQL Schema Compare Scripting Options

Scripting options allow you to control the way objects are scripted in the database synchronization script. You can choose, for example, not to script the table change tracking, script the start and end value of a sequence, omit the filegroup clause of database objects and many more.

> If you are not sure what options would be best for your scenario, click **Restore Defaults**, and then use the default options.

Most of the scripting options are self-explanatory, and for those that are not, the description provided in the Comparison Options window is sufficient. However, there are two options that merit special attention:

- **Script New Columns As NULL**. When the synchronization process requires adding a NOT NULL column on the target database, but no default constraint is specified for that column, SQL Server rejects the new column and the synchronization fails. In such scenarios the best course of action would be to add a default constraint on the source column. If such an action is not possible, you can add the new column to the target database as a NULL ALLOWED, and then, after the synchronization, connect to the target database via SSMS and change that particular column from NULL ALLOWED to NOT NULL.
- **Script New Constraints WITH NOCHECK**. When the synchronization requires adding new constraints, such as a foreign key or a check constraint, on a table, the existing data may violate those constraints, in which case SQL Server rejects the action and the synchronization script fails. To overcome this obstacle, you can choose to add those constraints the WITH NOCHECK clause first, and then verify if the existing data violates the constraints. When this option is ON, Schema Compare generates the following two synchronization scripts which are executed one after the other:
  - **Main synchronization script**. Creates the new constraints with NOCHECK so that no data violation occurs and databases are synchronized successfully.
  - **Constraints enabling script**. Enables the constraints that were created WITH NOCHECK and runs after the main script, on a separate transaction. If the existing data violates these constraints, then the script fails, but the failure does not affect the main database synchronization. In a case of failure, you must address the constraint violations on a case-by-case basis.
- **Creating CLR assemblies on SQL Server 2017 or higher**. Starting with SQL Server 2017, the code access security in .NET framework is no longer used as a security boundary for CLR assemblies. A new configuration option, named "*clr strict security*" has been added to enhance the security of CLR assemblies. When this option is ON, which is the default value, SQL Server treats all assemblies, even those marked with SAFE or EXTERNAL_ACCESS, as if they were UNSAFE.
  With this new policy in place, simply creating an assembly, as it was done prior to SQL Server 2017, may not be sufficient. Therefore, starting with version 10, schema compare provides the following scripting options, so that assemblies complies with the new SQL Server policy and are registered successfully.

1. **Add assembly to the trusted assembly list**. White-list the assembly by adding it to the trusted assembly list via the SQL Server 2017 sp_add_trusted_assembly. You must be a sysadmin or have CONTROL SERVER permission to execute sp_add_trusted_assembly.
2. **Set the database TRUSTWORTHY ON**. (Not recommended) Generates a statement that alters the database and enables the TRUSTWORTHY setting. You must be a sysadmin to enable this setting.
3. **Disable "CLR strict security" option**. (Not recommended) Generates a statement that disables the SQL 2017 option "CLR strict security". You must have ALTER SETTINGS server-level permission to change this option. ALTER SETTINGS permission is implicitly held by the sysadmin and serveradmin fixed server roles.
4. **None of the above**. Use this option if you've already configured the database to accept CLR assemblies. Microsoft, for example, recommends the followings:
    - Sign the assembly with a certificate or a strong-name.
    - Create a SQL Server login for the assembly certificate or the asymmetric key contained in the strong-name signing file.
    - Grant UNSAFE ASSEMBLY permission to the login in the master database.

For more information on CLR assemblies, check the SQL Server "clr strict option" https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/clr-strict-security?view=sql-server-ver15.