

# Instruments for working with database

Backward compatible SQL scripts

Version 8.0



This documentation is provided under restrictions on use and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this documentation, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

# Table of Contents

<b>Backward compatible SQL scripts</b>	<b>4</b>
Usage recommendations for backward compatible SQL scripts	4
Implement a backward compatible SQL script	5
Development recommendations for backward compatible SQL scripts	5

# Backward compatible SQL scripts



Medium

Backward compatible SQL scripts are available in Creatio version 8.0.3 and later.

If the package installation ends with an error, you cannot roll the configuration back completely since SQL scripts make irreversible changes to the database as part of the package installation. To prevent that, use backward compatible SQL scripts instead.

A **backward compatible SQL script** (safe SQL script) is an SQL script whose execution does not make irreversible changes to the database. Changes made by backward compatible SQL scripts as part of the package installation do not affect Creatio's operability. Use backward compatible SQL scripts to ensure you can roll back to the previous configuration state completely in case of an error. Learn more in a separate article: [Delivery management process](#).

**Note.** Since you can roll back to the previous configuration state from the package backup after the package installation is complete, restoration is accomplished by re-executing the previous version of the SQL script rather than by canceling the transaction.

Creatio version 8.0.5 Atlas and later lets you customize delivery process to execute CRUD operations with data as part of package installation in a way that does not cause irreparable changes to the database. Learn more in a separate article: [Customize delivery process](#).

To develop a backward compatible SQL script, follow the recommendations listed in a different section during the development: [Development recommendations for backward compatible SQL scripts](#).

## Usage recommendations for backward compatible SQL scripts

Creatio lets you roll back to the previous configuration state from the package backup if the package installation ends with an error or a successful installation results in issues with Creatio functionality. Learn more in a separate article: [Delivery management](#). To successfully roll back to the previous configuration state from the package backup, [ *SQL script* ] configuration items must meet the guidelines.

**Recommendations** to ensure successful rollback to the previous configuration state from the package backup:


- No new or changed package configuration elements of the [ *SQL script* ] type. I.e., the modification date of the configuration element of the [ *SQL script* ] type is the same as date of its modification on the production environment to install the package.
- New or changed package configuration elements of the [ *SQL script* ] type are backward compatible.

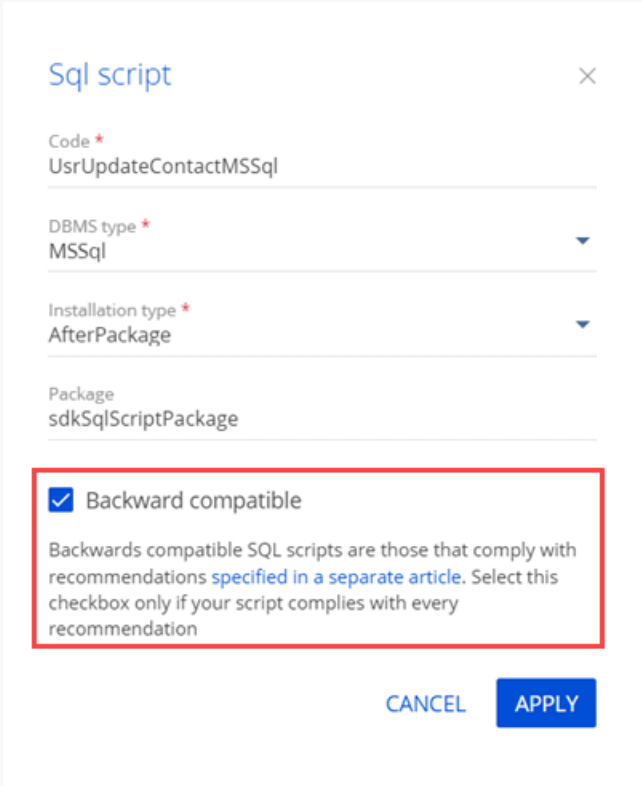
Compliance with one of the recommendations ensures successful rollback to the previous configuration state from the package backup.

## Implement a backward compatible SQL script

1. Create a configuration element of the [ *SQL script* ] type. To do this, follow the instructions in a separate article: [SQL script](#).
2. Implement a script that complies with the recommendations. Learn more about the recommendations in a different section: [Development recommendations for backward compatible SQL scripts](#).

Follow these **practices** when you develop a backward compatible SQL script:

- Analyze the potential behavior of Creatio after it executes the SQL script.
  - Take into account the way the SQL script affects Creatio operability after the configuration rollback.
3. Click  in the setup area of the Script Designer and select the [ *Backward compatible* ] checkbox. The checkbox flags the SQL scenario as backward compatible. The developer that implements the SQL scenario is in charge of selecting the checkbox and ensuring the SQL script complies with the [recommendations](#).



**Sql script** [X]

Code \*  
UsrUpdateContactMSSql

DBMS type \*  
MSSql

Installation type \*  
AfterPackage

Package  
sdkSqlScriptPackage

Backward compatible

Backwards compatible SQL scripts are those that comply with recommendations [specified in a separate article](#). Select this checkbox only if your script complies with every recommendation

CANCEL APPLY

**Attention.** If you install a package that contains at least one configuration element of the [ *SQL script* ] type without the [ *Backward compatible* ] checkbox selected, fails, rollback to the previous configuration state from the package backup via the WorkspaceConsole utility will be suspended. Learn more in another article: [Delivery management](#).

## Development recommendations for backward compatible SQL scripts

Follow the development **recommendations** to create backward compatible SQL scripts that do not cause issues with Creatio functionality after the configuration rollback. The recommendations are based on open sources as well as the official documentation of Microsoft SQL, PostgreSQL, and Oracle database management systems supported by Creatio.

**Recommendation.** Follow the ACID principle when you develop SQL scripts.

The **ACID principle** is a set of properties that ensures the reliability of database transactions. Learn more about the ACID principle in [Wikipedia](#).

The ACID principle is based on the following **properties**:

- **Atomicity**
- **Consistency**
- **Isolation**
- **Durability**

A single SQL script must correspond to a single entity. To **execute several SQL scripts in a set order**, follow the instructions in a separate article: [SQL script](#).

**Recommendation.** Do not use SQL scripts to change database entities.

A database entity is a view, function, procedure, or trigger. When you develop SQL scripts that change entities, keep in mind that Creatio might not recognize certain database tables or database table fields when you roll back to the previous configuration state from the package backup. Use SQL scripts to recreate entities instead of editing them.

To **recreate an entity using an SQL script**:

1. Make sure the needed entity exists in the database tables.
2. Delete the current entity version.
3. Create a new entity version. Use the `CREATE` operator instead of `ALTER` operator to do this.
4. Set the [ *Installation type* ] property of the configuration element of the [ *SQL script* ] type to "AfterPackage" or "AfterSchemaData." Learn more about the available values of the [ *Installation type* ] property in a separate article: [SQL script](#).

**Recommendation.** Do not use SQL scripts to create triggers.

For example, if you develop an SQL script that creates a field management trigger, Creatio behaves as follows. After Creatio executes an SQL script and you roll back to the previous configuration version from the package backup, issues with Creatio functionality can occur. Since the fields created by the trigger are new for the version to which you rolled back, you cannot delete the trigger. As such, this can lead to issues with Creatio functionality.

**Recommendation.** Do not use SQL scripts to delete database entities.

A database entity is a view, procedure, function, trigger, or database table.

We do not recommend deleting unused entities. Deleting unused entities violates dependencies in previous versions, thus causing issues with Creatio functionality. After Creatio executes the SQL script and you roll back to the previous configuration version from the package backup, deleted entities are not restored unless the SQL script recreates them in the previous package version. If you decide to delete a database entity regardless, make sure Creatio has not used it for a significant time.

Situations when you need to delete entities when deleting an app from Creatio, for example, when deleting Marketplace apps, are an exception.

To **delete an entity using an SQL script when deleting an app**, set the [ *Installation type* ] property of the configuration element of the [ *SQL script* ] type to "UninstallApp." Learn more about the available values of the [ *Installation type* ] property in a separate article: [SQL script](#).

**Recommendation.** Do not use SQL scripts to create and edit database tables.

Use **objects** instead of SQL scripts to create and edit database tables. Learn more in a separate article: [Object](#).

You can use SQL scripts to create and edit the following **database tables**:

- temporary tables
- service tables, i. e., tables that do not contain business data and are required to solve technical problems

Follow these **practices** when creating and editing such database tables using SQL scripts:

- Make sure changes described by the SQL script do not cause issues with Creatio functionality.
- Use table versioning that lets you switch logic between table versions.
- Make sure fields, tables, and other objects required for the SQL scenario exist.
- Keep in mind the use of such tables and their fields in indexes, primary keys, procedures, functions, triggers, and views.

**Recommendation.** Do not use SQL scripts to change the field type or requirement flag in database tables.

Create new fields using **objects** to change the field type or requirement flag in database tables instead of changing the existing fields using SQL scripts. Learn more in a separate article: [Object](#).

For example, if you execute an SQL script that changes the field type, Creatio behaves as follows. After Creatio executes the SQL script and you roll back to the previous configuration version from the package backup, the mismatched data type error occurs.

For example, if you execute an SQL script that changes the field requirement flag, Creatio behaves as follows. After Creatio executes the SQL script and you roll back to the previous configuration version from the package backup, the field remains required and you still have to fill it out to save the record.

**Recommendation.** Do not use SQL scripts to add required fields that have default values to database

tables.

Use **objects** instead of SQL scripts to add required fields that have default values. Learn more in a separate article: [Object](#).

**Recommendation.** Do not use SQL scripts to add data to database tables.

Use **data bindings** or install scripts instead of SQL scripts to add data to database tables. Learn more in separate articles: [Packages basics](#), [Customize delivery process](#).

Make sure database tables do not have data that must be added using a complex filter or special procedure. This is relevant for key fields first and foremost. Creatio can execute an SQL script multiple times as part of debugging. Learn more in a separate guide: [Debugging](#). Creatio does not delete data added by the SQL script when you roll back to the previous configuration version from the package backup. This can lead to issues with Creatio functionality.

**Recommendation.** Do not use SQL scripts to change database table data.

Use **data bindings** or install scripts instead of SQL scripts to change database table data. Learn more in separate articles: [Packages basics](#), [Customize delivery process](#).

Some changes to database table data executed using an SQL script cannot be reversed using another SQL script or via a rollback to the previous configuration version from the package backup. For example, if Creatio executes an SQL script that changes the value of the `[IsActive]` column in the `[SysProcess]` database table from `true` to `false`, Creatio behaves as follows.

```
UPDATE SysProcess SET IsActive = true WHERE IsActive = false
```

After Creatio executes the SQL script, the `[IsActive]` column of the `[SysProcess]` database table has no `false` values. I. e., Creatio cannot determine which column values the SQL script changed. You can develop an SQL script that reverses the changes, but this requires significant effort on your end.

**Recommendation.** Do not use SQL scripts to delete business data from database tables.

We do not recommend deleting business data. For example, if you execute an SQL script that deletes field data, Creatio behaves as follows. After Creatio executes the SQL script and you roll back to the previous configuration version from the package backup, the field data remains deleted. I. e., the script deletes data permanently. The fact that the SQL script of the package to the version of which you roll back added this data is irrelevant. Use **objects** or install scripts instead of SQL scripts to create and change database table data. Learn more in separate articles: [Object](#), [Customize delivery process](#).

Service data, for example, the cleanup of the scheduler queue, is an exception. However, keep in mind that the deletion is permanent.



**Recommendation.** Ensure that the changes made by backward compatible SQL scripts do not affect the data integrity.

**Recommendation.** Check whether database tables contain procedures and functions before you call them in SQL scripts.

In most cases, you do not need to call procedures and functions in SQL scripts during a package installation. If such a need arises, make sure that the database tables contain the required procedures and functions.