Creatio Academy

Delivery

Delivery in WorkspaceConsole

Version 8.0

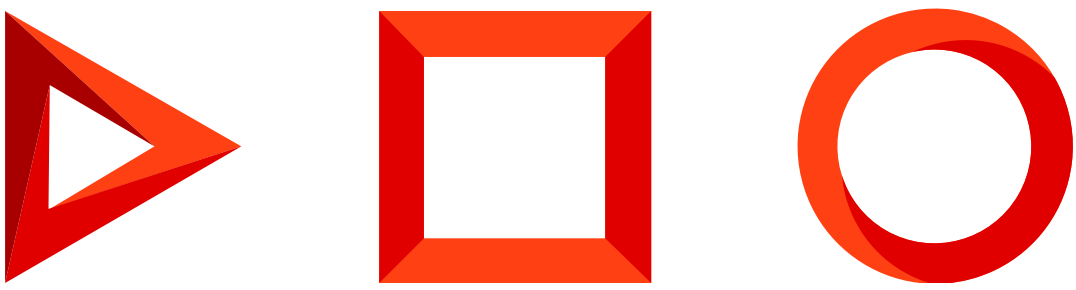


Table of Contents

Delivery in WorkspaceConsole

Medium

Creatio provides various tools for functionality delivery.

The **delivery management tools** are as follows:

- Creatio IDE
- WorkspaceConsole utility

This article covers supply management using the WorkspaceConsole utility.

WorkspaceConsole is a utility for operations with Creatio packages and schemas of configuration elements (client modules, objects, and source code (C#).

The WorkspaceConsole utility provides the following solution **transfer options**:

- Transfer packages and configuration element schemas between environments and configurations.
- Install new packages when updating or when exporting from a development environment.
- Transfer package-related data, such as the lookup content, new system settings, demo section records, etc.
- Transfer localizable resources.
- Create and transfer workspaces between environments.

WorkspaceConsole on .NET Framework

Set up the WorkspaceConsole utility before using it.

Set up the WorkspaceConsole utility

1. Find out the relevant connection string value.

   To do this, open the `..\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\Terrasoft.Tools.WorkspaceConsole.exe.config` file. The `connectionStringName` attribute of the `<db>` XML element contains the connection string.

   `Terrasoft.Tools.WorkspaceConsole.exe.config` file

   ```
   <terrasoft>
       ...
       <db>
           <general connectionStringName="db" securityEngineType="Terrasoft.DB.MSSql.MSSqlSecuri
       </db>
       ...
   </terrasoft>
   ```

2. Edit the connection string.

   To do this, open the `ConnectionStrings.config` file in the Creatio root directory. The `name` attribute of the `<connectionStrings>` XML element contains the connection string. The value of the `name` attribute in the `ConnectionStrings.config` file must match the value of the `connectionStringName` attribute in the `Terrasoft.Tools.WorkspaceConsole.exe.config` file.

   `ConnectionStrings.config` file

   ```
   <connectionStrings>
       <add name="db" connectionString="Data Source=dbserver\MSSQL2016; Initial Catalog=YourDBNa
       <add name="dbOracle" connectionString="Data Source=(DESCRIPTION = (ADDRESS_LIST = (ADDRES
   </connectionStrings>
   ```

   The configuration file has the following connection string **types** set up:

   - `"db"` string. Required to connect to the Microsoft SQL server database.
   - `"dbOracle"` string. Required to connect to the Oracle database.

   To perform a one-time operation with WorkspaceConsole, run the utility with the `-webApplicationPath` parameter. Specify the path to the Creatio directory in this parameter. In this case, the utility independently determines all necessary database connection settings from the `ConnectionStrings.config` file. However, the connection parameters in the `Terrasoft.Tools.WorkspaceConsole.exe.config` file are ignored.

3. Enable downloading from remote sources. To do this, set the `enabled` attribute of the `loadFromRemoteSources` element in the `Terrasoft.Tools.WorkspaceConsole.exe.config` file to `true`.

   ```
   <loadFromRemoteSources enabled="true" />
   ```

4. Install the WorkspaceConsole utility.

   To do this, run a preinstalled batch command file in the `..\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\` directory as an administrator. This is required to install the executable file version and libraries the utility uses.

   Batch files of the WorkspaceConsole utility commands:

   - For **32-bit operating systems**, run the `PrepareWorkspaceConsole.x86.bat` file.
   - For **64-bit operating systems**, run the `PrepareWorkspaceConsole.x64.bat` file.

5. Configure the utility to perform operations on the SVN repository (optional).

   To do this, copy the `SharpPlink-x64.svnExe`, `SharpSvn.dll`, and `SharpSvn-DB44-20-x64.svnDll` files from the appropriate directory to the `...\Terrasoft.WebApp\DesktopBin\WorkspaceConsole` directory.

   - For **32-bit operating systems**, copy the files from the `...\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\x86` directory.
   - For **64-bit operating systems**, copy the files from the `...\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\x64` directory.

The `Terrasoft.Tools.WorkspaceConsole.exe` executable file of the utility is located in the `..\Terrasoft.WebApp\DesktopBin\WorkspaceConsole` directory. The utility version matches the Creatio version.

Attention. The WorkspaceConsole utility version must match the Creatio version. For example, if the current Creatio version is 7.18.1.1794, and you need to update the packages to version 7.18.2.1658, use WorkspaceConsole version 7.18.2.1658. To obtain the relevant version of the utility, contact support.

WorkspaceConsole works directly with the Creatio database. Thus, specify the database information in the `Terrasoft.Tools.WorkspaceConsole.exe.config` utility configuration file to ensure the utility works properly. If you deploy Creatio in the cloud, only employees of the cloud service department can work with the utility. In this case, contact support to transfer the changes.

We recommend preparing the `WorkspaceConsole` commands in a batch file (*.bat or *.cmd) created in a text editor.

WorkspaceConsole utility lets you perform the following **actions**:

1. Check data bindings.
2. Back up the database.
3. Export packages.
4. Import packages.
5. Change the feature status.
6. Restore the configuration from the package backup.
7. Update the configuration.
8. Delete the packages.

Restart Creatio in IIS to apply the changes. This is required since the WorkspaceConsole utility makes changes directly in the database.

Check data bindings

Before exporting the package, check that the data is bound to the package correctly. Bound data includes lookup values, new system settings, demo section records, etc.

If you create a section in the Section Wizard, the Wizard automatically binds the data required for the section to work. To ensure Creatio displays the section in the workplace after the import, bind the corresponding value of the `SysModuleInWorkplace` object.

Back up the database

Before you implement the changes in Creatio using the WorkspaceConsole utility, back up the database. This lets you restore Creatio if the utility commands and parameters are used incorrectly.

Export packages

Use the WorkspaceConsole utility to export a package from the database or SVN repository.

Export packages from the database

1. Create the command to export packages from the database.

   Run the command to export packages from the database at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=SaveDBContent -con
```

WorkspaceConsole parameters for exporting packages

Parameter	Value	Description
`-operation`	`SaveDBContent`	Saves the database content to the file system. The `-contentTypes` parameter determines the content type. The `-destinationPath` parameter determines the content export path in the file system. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`-contentTypes`	`Repository`	The type of content exported from the database to the drive. If set to `Repository`, the utility exports the workspace specified in the `-workspaceName` parameter to the directory specified in the `-destinationPath` parameter.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration) where the operation is performed. By default, all users work in the `Default` workspace.
`-destinationPath`	[*Path to a local directory*]	Path to a local directory. The utility exports the *.gz package archives to this directory.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.

2. Run the utility.

Note. The package export operation exports all packages of the workspace. The process might take several dozen minutes.

As a result, the utility will export the workspace package archives to the local directory.

Export packages from the SVN repository

1. Create the command to export packages from SVN.

   Run the command to export packages from the SVN repository at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=SaveVersionSvnCont
```

WorkspaceConsole parameters for exporting packages

Parameter	Value	Description
`-operation`	`SaveVersionSvnContent`	Downloads the package hierarchy as a set of *.zip archives. The `-destinationPath` parameter determines the content export path in the file system. The `-sourcePath` parameter determines the SVN repositories.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration) where the operation is performed. By default, all users work in the `Default` workspace.
`-destinationPath`	[*Path to a local directory*]	Path to a local directory. The utility will export the *.gz package archives to this directory.
`-workingCopyPath`	[*Path to a local directory*]	The local directory for the working copies of packages stored in SVN.
`-sourcePath`	[*Path to the SVN repository*]	The address of the SVN repository that stores the package structure and metadata. Can accept multiple values separated by commas.
`-packageName`	[*Package name*]	The name of the package from the source SVN repository. The utility downloads the

		packages on which the current package depends as well.
`-packageVersion`	[*Package version*]	The version of the package from the source SVN repository.
`-sourceControlLogin`	[*SVN username*]	SVN user login.
`-sourceControlPassword`	[*SVN user password*]	SVN user password.
`-cultureName`	[*Language culture*]	Language culture code. For example, `en-US`.
`-excludeDependentPackages`	`true` or `false`	The flag that specifies whether to export packages on which the package listed in the `-packageName` parameter depends.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.

2. Run the utility.

As a result, the utility will export the workspace package archives to the local directory.

Import packages

1. Create the command to import packages into the database.

   Run the command to import packages into the database at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=InstallFromReposit
```

WorkspaceConsole parameters for importing packages

Parameter	Value	Description
`-operation`	`InstallFromRepository`	Imports the package content and metadata from *.zip archives into the configuration. If needed, the utility runs bound SQL scripts, re-generates source code, and installs bound data. Works only with updated or new packages and their elements. Requires either the `-webApplicationPath` or

		`-configurationPath` parameter. Also requires the `-confRuntimeParentDirectory` parameter.
`-packageName`	[*Package name*]	In the `-workspaceName` parameter, specify the package name in the configuration. The utility downloads the packages on which the current package depends as well. If you omit the parameter, the utility downloads all packages of the configuration.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration) where the operation is performed. By default, all users work in the `Default` workspace.
`-sourcePath`	[*Path to a local directory*]	Path to a local directory. The directory must contain the *.gz package archives to install.
`-destinationPath`	[*Path to a local directory*]	Path to a temporary local directory. The utility unpacks the package archives specified in the `-sourcePath` parameter to this directory.
`-skipConstraints`	`false`	Skips the creation of foreign keys in database tables. Can be `true` or `false`.
`-skipValidateActions`	`true`	Skips checking if table indexes can be created when updating the database structure. Can be `true` or `false`.
`-regenerateSchemaSources`	`true`	Specifies whether to re-generate the source code after saving the packages to the database. Can be `true` or `false`. The default value is `true`.
`-updateDBStructure`	`true`	Specifies whether to update the database structure after saving the packages. Can be `true` or `false`. The default value is `true`.
`-updateSystemDBStructure`	`true`	Specifies whether to modify the structure of the system schema database before installing the packages.

		Also creates the missing indexes in the system tables. Can be `true` or `false`.
`-installPackageSqlScript`	`true`	Specifies whether to run the SQL scripts before and after saving the packages. Can be `true` or `false`. The default value is `true`.
`-installPackageData`	`true`	Specifies whether to install the bound package data after saving the packages. Can be `true` or `false`. The default value is `true`.
`-continueIfError`	`true`	Specifies whether to abort the installation after encountering the first error. If set to `true`, the installation process goes through to the end. You will receive the list of the errors encountered. Can be `true` or `false`. The default value is `false`.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.
`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.
`-backupConfiguration`	`true`	Specifies whether to back up the configuration before package

		configuration before package installation. Available in Creatio version 8.0.2 and later. The default value is `true`.

2. Run the utility.
3. Create the command to generate the static content in the file system.

   Run the command to generate the static content in the file system at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=BuildConfiguration
```

WorkspaceConsole parameters for generating the static content

Parameter	Value	Description
`-operation`	`BuildConfiguration`	Generates the static content in the file system. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration) where the exported packages are specified. By default, all users work in the `Default` workspace.
`-destinationPath`	[*Path to a local directory*]	Path to a local directory. The utility will export the *.gz package archives specified in the `-sourcePath` parameter to this directory.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.

`-force`	`true` or `false`	Sets the conditions of the file content generation If set to `true`, the utility generates the file content for all schemas. If set to `false`, the utility generates the file content for updated schemas. The default value is `false`. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.

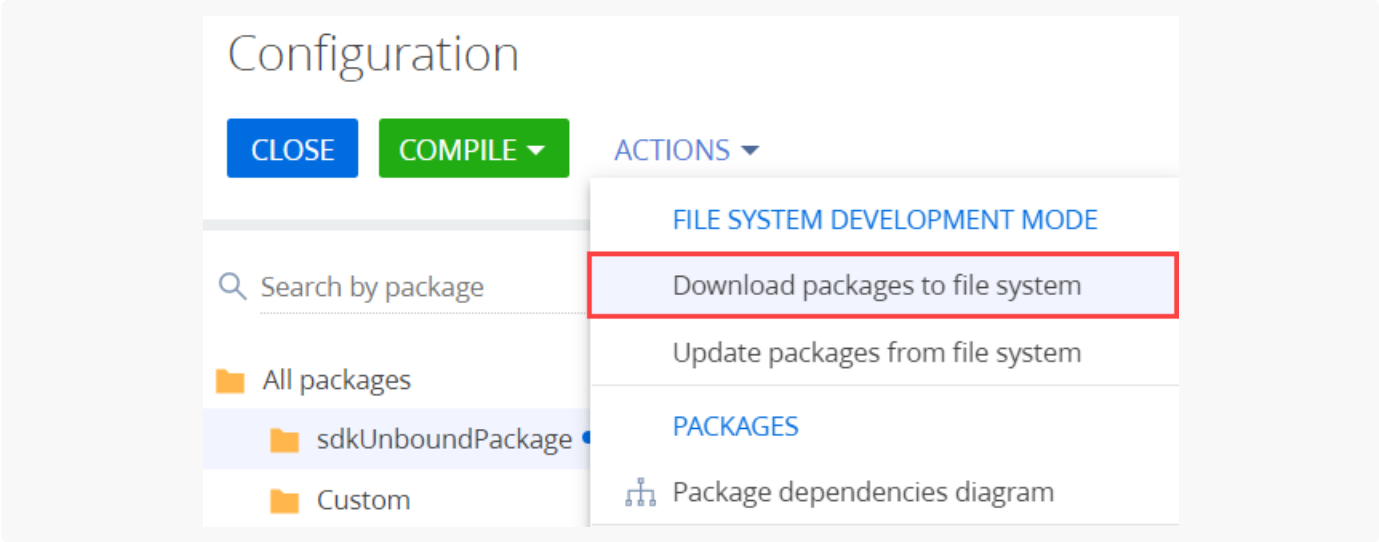
4. Run the utility.

Note. Creatio considers packages imported using the WorkspaceConsole as preinstalled. You cannot modify them.

We do not recommend using the WorkspaceConsole utility for importing packages into the database if the **file system development mode is enabled**. If you use the utility, it modifies the schema source code in the database but not in the file system. As such, if you open the configuration element schema in Creatio IDE, the schema will contain unmodified code from the file system. However, the modification date of the configuration element schema will be updated. This will make the schema falsely appear to have transferred correctly.

To import packages into the database when the **file system development mode is enabled**:

1. Use the WorkspaceConsole utility to import packages into the database. To do this, follow this guide: Import packages.
2. Select [*Download packages to the file system*] in the [*File system development mode*] action group on the toolbar.



As a result, Creatio will download all packages to the `..\Terrasoft.WebApp\Terrasoft.Configuration\Pkg` path as a set of directories that match package names.

Change the feature status

Learn more about feature in a separate article: Manage existing feature.

Since version 8.0.4, Creatio lets you manage the feature status using the WorkspaceConsole utility.

To **change the feature status**:

1. Create the command to change the feature status.

   Run the command to change the feature status at the command line interface (Windows command prompt). Structure the command as follows:

   ```
   [Path to WorkspaceConsolee]\Terrasoft.Tools.WorkspaceConsole.exe -operation=ChangeFeatureStat
   ```

WorkspaceConsole parameters for changing feature status

Parameter	Value	Description
-operation	ChangeFeatureState	Changes the feature status. Requires either the -featureCode, -featureState or -adminUnitName parameters.
-featureCode	[*Feature code*]	The code of the additional feature. If you change the feature absent from the database, the utitily adds the corresponding record to the database table automatically.
-featureState	[*Feature status*]	The status of the additional feature (true — turned on, false — turned off).
-adminUnitName	[*User name*]	The name of the user (the [Name] column value in the [SysAdminUnit] database table) that changes the feature status. If you specify the parameter, the utility modifies the corresponding record in the [AdminUnitFeatureState] database table. If you omit the parameter, the utility sets the default value for the feature (the [DefaultState] column value in the [Feature] database table).
-workspaceName	[*Workspace name*]	The name of the workspace (configuration) where the deleted packages are specified. By default, all users work in the Default workspace.
-confRuntimeParentDirectory	[*Path to a local directory*]	Path to the parent directory of the ..\Terrasoft.WebApp\conf directory.
-logPath	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.
-autoExit	true or false	Specifies whether to finish the utility process automatically after performing the operation. Can be true or false. The default value is false.

2. Run the utility.

As a result, Creatio will change the status of the additional feature.

Restore the configuration from the package backups

Take this step if package import results in an error. Since version 8.0.2, Creatio lets you restore the configuration from the package backups using the WorkspaceConsole utility. The mechanism works similarly to the functionality that restores the configuration from backup available when installing apps or packages using custom means.

To **restore the configuration from the package backups**:

1. Create the command to restore the configuration from the package backups.

   Run the command to restore the configuration from the package backups at the command line interface (Windows command prompt). Structure the command as follows:

   ```
   [Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=RestoreConfigurati
   ```

WorkspaceConsole parameters for restoring the configuration from the package backups

Parameter	Value	Description
`-operation`	`RestoreConfiguration`	Restores the configuration from the package backups. Requires the `BackupConfiguration` operation to be performed. Requires the `-backupPath` parameter.
`-backupPath`	[*Path to a local directory*]	Path to a local directory that contains the *.gz archives of package backups. The value of the current parameter must match the value of the `-backupPath` parameter when backing up packages.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory.
`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.

2. Run the utility.

As a result, Creatio will restore the configuration from the package backups.

Update the configuration

Since version 8.0.4, Creatio lets you update the configuration using the WorkspaceConsole utility. You can install and delete packages as part of an update. If you update the configuration, Creatio backs up the installed and deleted packages.

You can perform the following **actions** as part of a configuration update:

- Roll back to the previous state of the installed packages.
- Restore the packages that were deleted as part of the configuration update.

To **update the configuration**:

1. Create the command to update the configuration.

   Run the command to update the configuration at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=UpdateConfiguratio
```

WorkspaceConsole parameters for updating the configuration

Parameter	Value	Description
`-operation`	`UpdateConfiguration`	Updates the configuration.
`-packageName`	[*Package name*]	In the `-workspaceName` parameter, specify the package name in the configuration. The utility downloads the packages on which the current package depends as well. If you omit the parameter, the utility downloads all packages of the configuration.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration). By default, all users work in the `Default` workspace.
`-sourcePath`	[*Path to a local directory*]	Path to a local directory. The directory must contain the *.gz package archives to install.
`-destinationPath`	[*Path to a local directory*]	Path to a local directory. The utility will export the *.gz package archives specified in the `-sourcePath` parameter to this directory.
`-skipConstraints`	`false`	Skips the creation of foreign keys in database tables. Can be `true` or `false`.
`-skipValidateActions`	`true`	Skips checking if table indexes can be created when updating the database structure. Can be `true` or `false`.
`-regenerateSchemaSources`	`true`	Specifies whether to re-generate the source code after saving the packages to the database. Can be `true` or `false`. The default value is `true`.
`-updateDBStructure`	`true`	Specifies whether to update the database

		structure after saving the packages. Can be `true` or `false`. The default value is `true`.
`-updateSystemDBStructure`	`true`	Specifies whether to modify the structure of the system schema database before installing the packages. Also creates the missing indexes in the system tables. Can be `true` or `false`.
`-installPackageSqlScript`	`true`	Specifies whether to run the SQL scripts before and after saving the packages. Can be `true` or `false`. The default value is `true`.
`-installPackageData`	`true`	Specifies whether to install the bound package data after saving the packages. Can be `true` or `false`. The default value is `true`.
`-packagesToDelete`	[*Packages to be deleted*]	The name of packages to delete. Can take multiple values separated by commas. Specify whether to delete the package as part of the configuration update.
`-continueIfError`	`true`	Specifies whether to abort the installation after encountering the first error. If set to `true`, the installation process goes through to the end. You will receive the list of the errors encountered. Can be `true` or `false`. The default value is `false`.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.

`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.

2. Run the utility.

As a result, Creatio will update the configuration.

Delete the packages

Since version 8.0.4, Creatio lets you delete one or more packages using the WorkspaceConsole utility.

To **delete the packages**:

1. Create the command to delete the packages.

   Run the command to delete the packages at the command line interface (Windows command prompt). Structure the command as follows:

```
[Path to WorkspaceConsole]\Terrasoft.Tools.WorkspaceConsole.exe -operation=DeletePackages -wo
```

WorkspaceConsole parameters for deleting packages

Parameter	Value	Description
`-operation`	`DeletePackages`	Deletes the packages from the configuration. Requires either the `-packagesToDelete` parameter.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration). By default, all users work in the `Default` workspace.
`-packagesToDelete`	[*Packages to be deleted*]	The name of packages to delete. Can take multiple values separated by commas. Specify whether to delete the package as part of the configuration update.
`-continueIfError`	`true`	Specifies whether to abort the installation after encountering the first error. If set to `true`, the installation process goes through to the end. You will receive the list of the errors encountered. Can be `true` or `false`. The default value is `false`.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. The file name consists of the operation start date and time. Optional.

2. Run the utility.

As a result, Creatio will delete the specified packages from the configuration.

WorkspaceConsole on .NET Core and .NET 6

You can use the WorkspaceConsole utility to generate and compile Creatio .NET Core or .NET 6.

To **generate and compile Creatio .NET Core or .NET 6**:

1. Edit the connection string.

   To do this, open the `Terrasoft.Tools.WorkspaceConsole.dll.config` file in the `..\WorkspaceConsole` directory. The `name` attribute of the `<connectionStrings>` XML element contains the connection string. The `"db"` string in the `Terrasoft.Tools.WorkspaceConsole.dll.config` file must match the corresponding string in the `ConnectionStrings.config` file.

   `Terrasoft.Tools.WorkspaceConsole.dll.config` file

   ```
   <connectionStrings>
      ...
       <add name="db" connectionString="Pooling=true; Database=7.18.4(netcore); Host=; Port=5432
       ...
   </connectionStrings>
   ```

   `"db"` string is required to connect to the Microsoft SQL Server database.

2. Open the command line and go to the `..\WorkspaceConsole` directory. To do this, run the command below.

   ```
   cd [Path to the WorkspaceConsole directory]
   ```

3. Configure commands to generate and compile Creatio .NET Core or .NET 6.

   Run commands to generate and compile Creatio .NET Core or .NET 6 at the command interpreter (console) sequentially. Structure the commands as follows:

   ```
   dotnet Terrasoft.Tools.WorkspaceConsole.dll -operation=RegenerateSchemaSources -workspaceName

   dotnet Terrasoft.Tools.WorkspaceConsole.dll -operation=RebuildWorkspace -workspaceName=Defaul

   dotnet Terrasoft.Tools.WorkspaceConsole.dll -operation=BuildConfiguration -force=True -worksp
   ```

   WorkspaceConsole parameters for generating and compiling Creatio .NET Core or .NET 6

Parameter	Value	Description
`-operation`	`RegenerateSchemaSources`	Re-generates the source code and compiles it. Requires the `-confRuntimeParentDirectory` parameter.
	`ReBuildWorkspace`	Recompiles the workspace (configuration). Use when developing schemas in Visual Studio. Requires the

		Requires the `-confRuntimeParentDirectory` parameter.
	`BuildConfiguration`	Generates the static content in the file system. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`-workspaceName`	[*Workspace name*]	The name of the workspace (configuration). By default, all users work in the `Default` workspace.
`-configurationPath`	[*Path to a local directory*]	Path to the `..\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.
`-confRuntimeParentDirectory`	[*Path to a local directory*]	Path to the Creatio parent directory.
`-logPath`	[*Path to a local directory*]	The utility saves the operation log file to this directory. Optional.
`-webApplicationPath`	[*Path to a local directory*]	Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the connection string in the utility's configuration file.
`-force`	`True` or `False`	Sets the conditions of the file content generation. If set to `True`, the utility generates and compiles the source code for all schemas. If set to `False`, the utility generates and compiles the source code for updated schemas. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`-destinationPath`	[*Path to a local*	Path to a local directory. The utility

`-destinationPath`	[*Path to a local directory*]	Path to a local directory. The utility exports the *.gz package archives to this directory.

Export packages from the database

Medium

Example. Export the packages of the `Default` workspace to a directory.

- `C:\creatio` is the Creatio installation directory.
- `C:\SavedPackages` is the package export directory.
- `C:\Logs` is the export directory of the operation log file.

1. Configure the command for the package export from the database

1. Create a *.bat or *.cmd Windows batch file in a text editor.
2. Add the command that runs the utility to the file.

Command that runs the utility

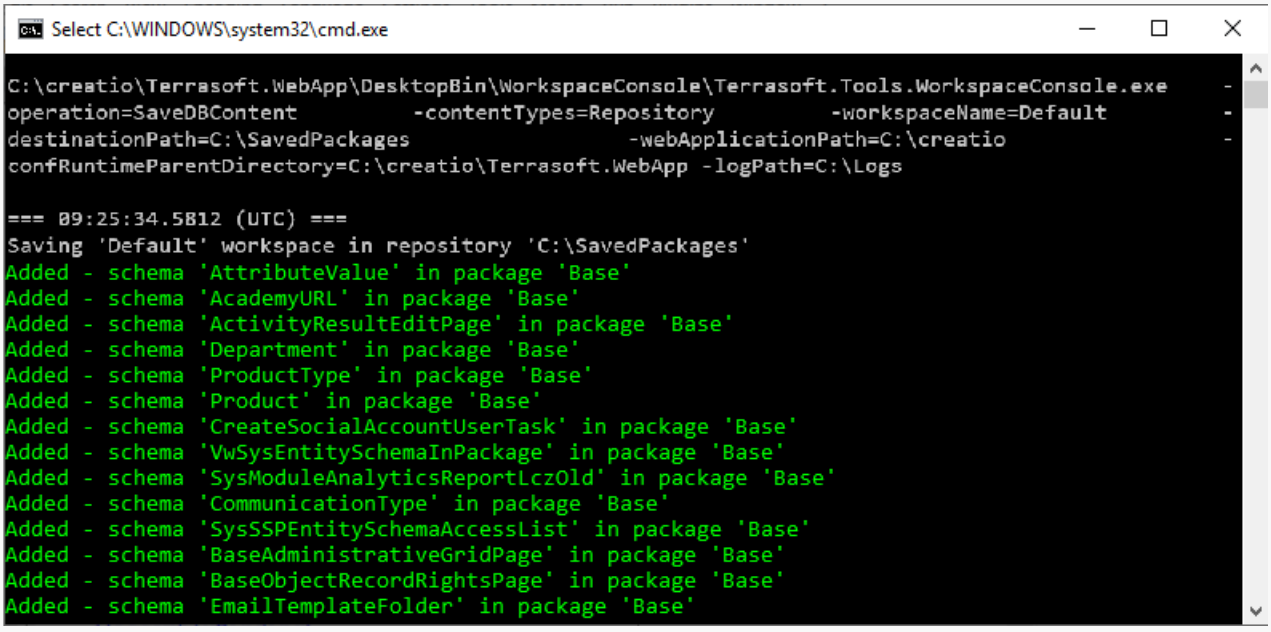
```
C:\creatio\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\Terrasoft.Tools.WorkspaceConsole.exe
pause
```

Save the batch file.

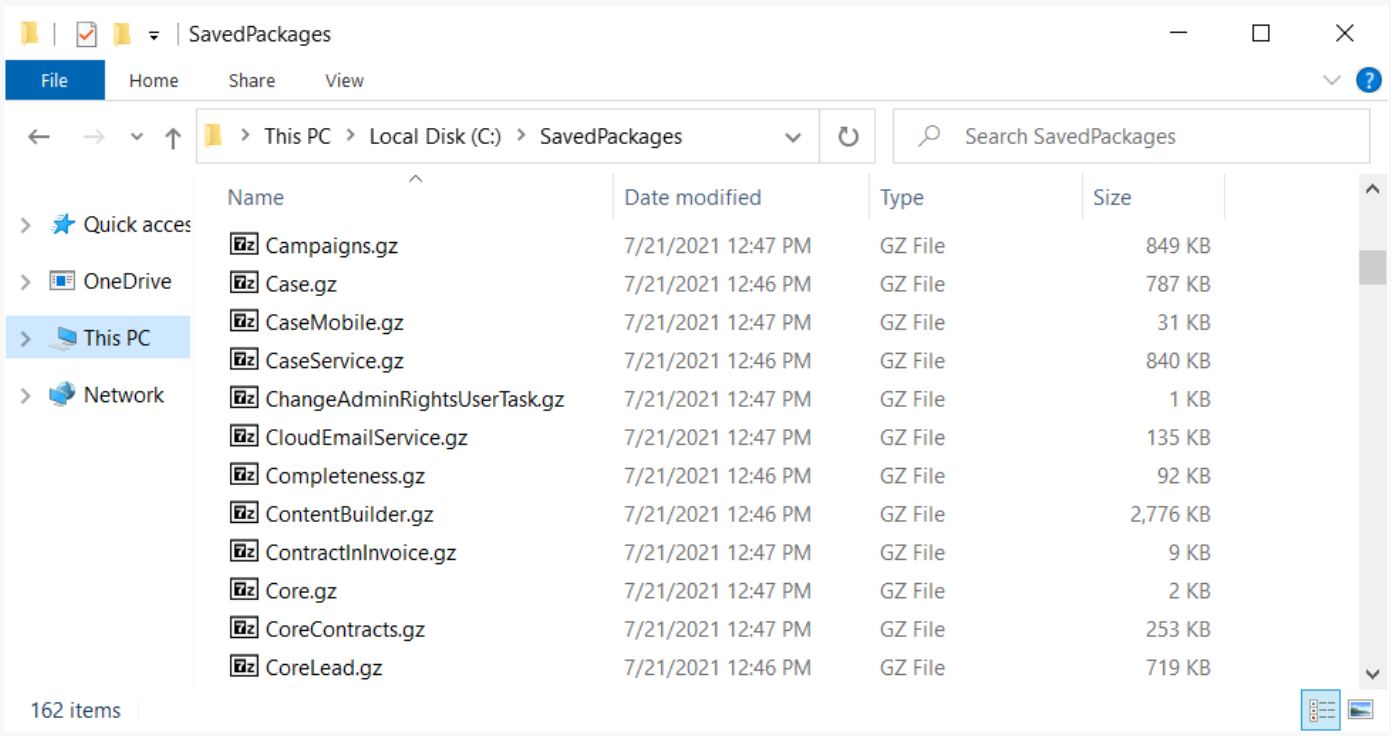
2. Export the package from the database

To **export the package from the database**, double-click the batch file name.

This will open the console that will output the execution process of the operation specified in the corresponding WorkspaceConsole command.



As a result, the utility will export the *.gz package archives of the `Default` configuration to the `C:\SavedPackages` directory.



Export a package from SVN

Medium

Example. Export the package from the SVN repository to the `Default` workspace directory.

- `C:\creatio` is the Creatio installation directory.
- `C:\SavedPackages` is the package export directory.
- `C:\WorkingCopy` is the export directory of the package structure in the SVN repository.
- `http://server-svn:8050/Packages` is the SVN repository URL.
- `sdkTestPackage` is the package to export from the SVN repository.
- `7.18.1` is the version of the package to export from the SVN repository.
- "`User`" is the login of the SVN repository user.
- "`Password`" is the password of the SVN repository user.
- `en-EN` is the language culture.
- `C:\Logs` is the export directory of the operation log file.

1. Configure the command for the package export from the SVN repository

1. Create a *.bat or *.cmd Windows batch file in a text editor.
2. Add the command that runs the utility to the file.

Command that runs the utility

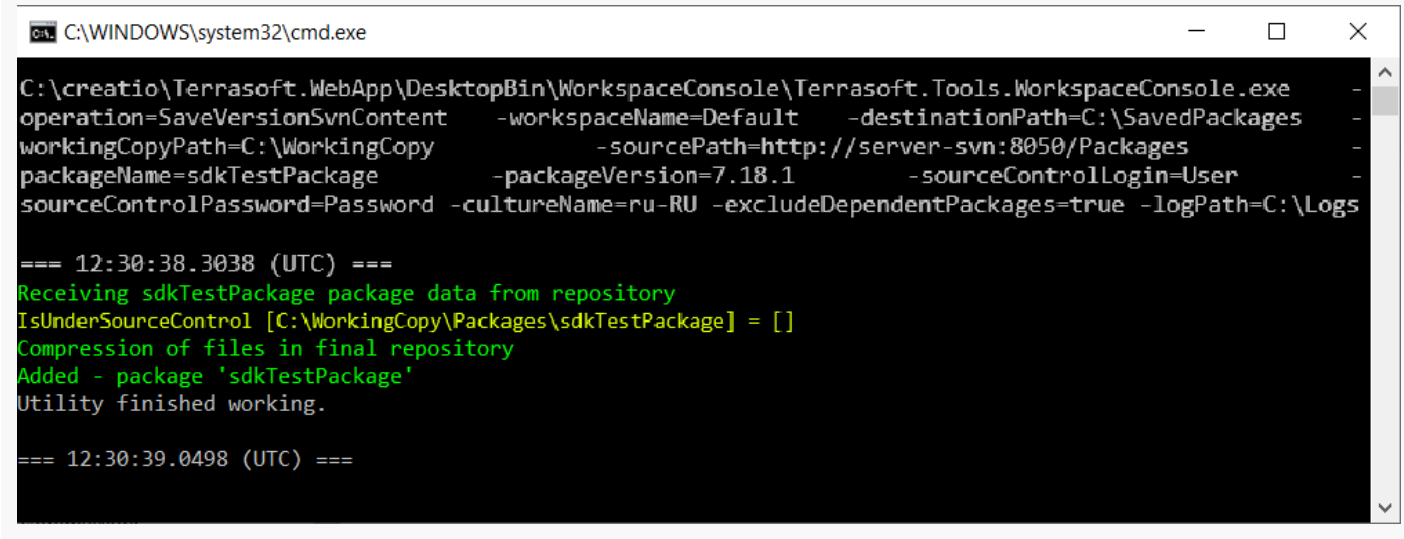
```
C:\creatio\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\Terrasoft.Tools.WorkspaceConsole.exe
pause
```

Save the batch file.

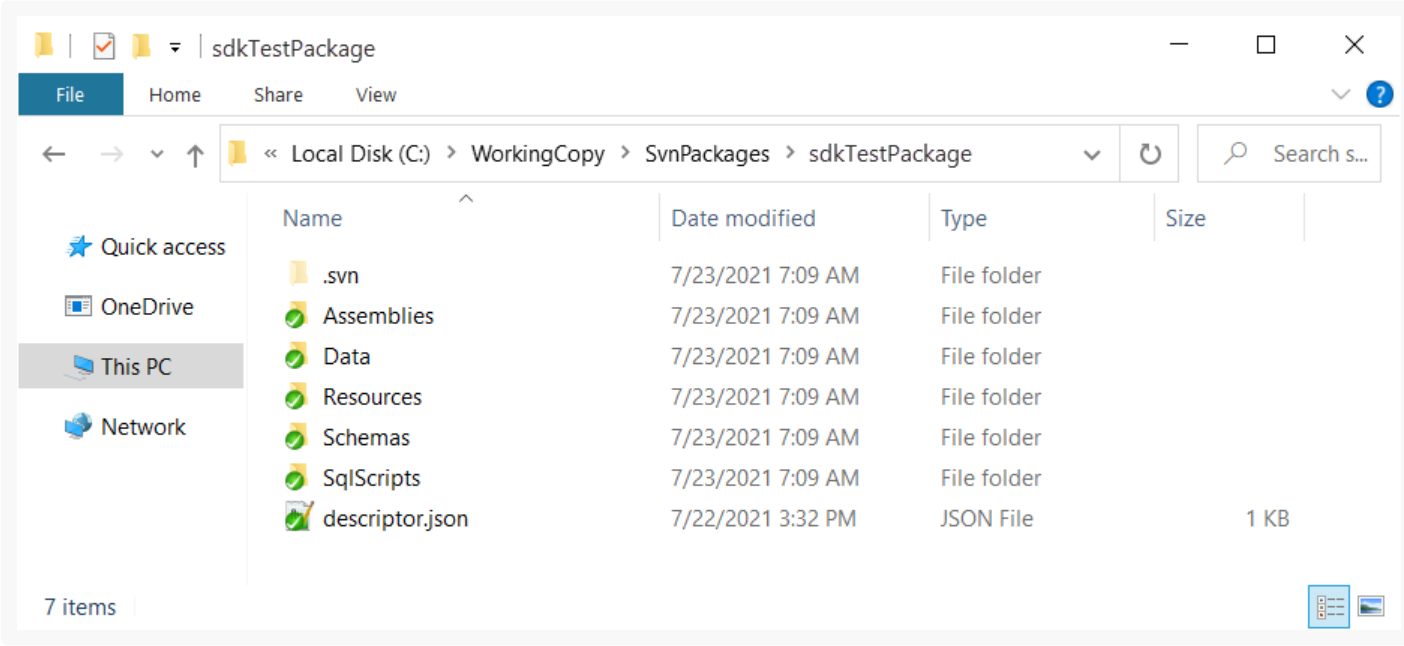
2. Export the package from SVN

To **export the package from SVN**, double-click the batch file name.

This will open the console that will output the execution process of the operation specified in the corresponding WorkspaceConsole command.



As a result, the utility will export the `sdkTestPackage` package of the `Default` configuration to the `C:\SavedPackages` directory. Learn more about the structure of the directory with the package name: Packages basics.



Import a package into the database

Medium

Example. Import the package from the directory into the `Default` workspace.

- `C:\creatio` is the Creatio installation directory.
- `sdkTestPackage` is the package to import into Creatio. The path to the package is `C:\SavedPackages`.
- `C:\SavedPackages` is the package directory.

- `C:\TempPackages` is the package import directory.
- `C:\Logs` is the export directory of the operation log file.

1. Configure the command to import the package into the database

1. Create a *.bat or *.cmd Windows batch file in a text editor.
2. Add the command that runs the utility to the file.

Command that runs the utility

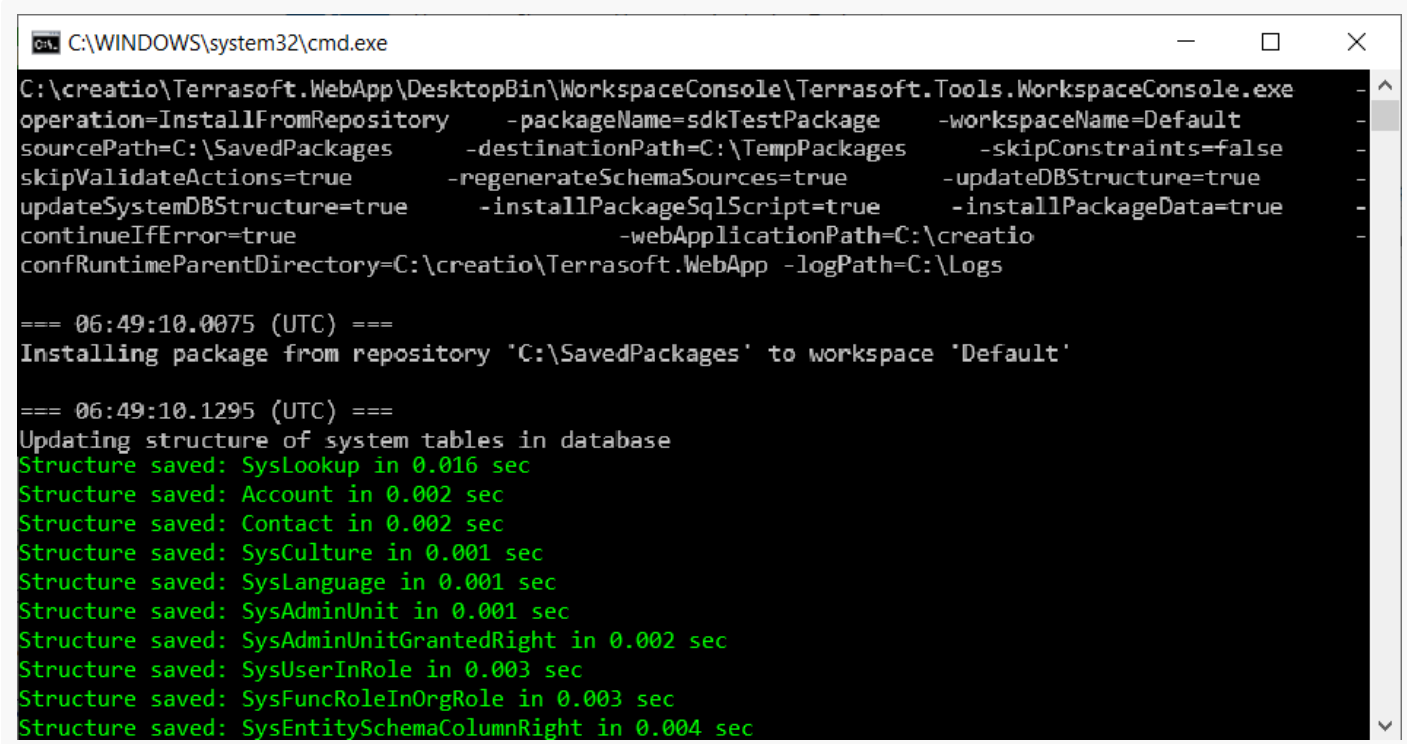
```
C:\creatio\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\Terrasoft.Tools.WorkspaceConsole.exe
pause
```

Save the batch file.

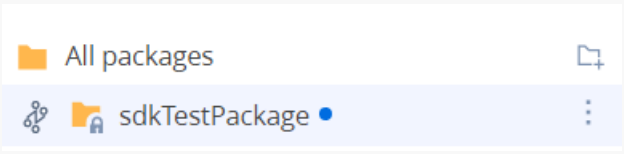
2. Import the package into the database

To **import the package into the database**, double-click the batch file name.

This will open the console that will output the execution process of the operation specified in the corresponding WorkspaceConsole command.



As a result, the utility will import the `sdkTestPackage` package into the `Default` configuration.



3. Configure the command to generate the static content in the file system

1. Create a *.bat or *.cmd Windows batch file in a text editor.
2. Add the command that runs the utility to the file.

Command that runs the utility

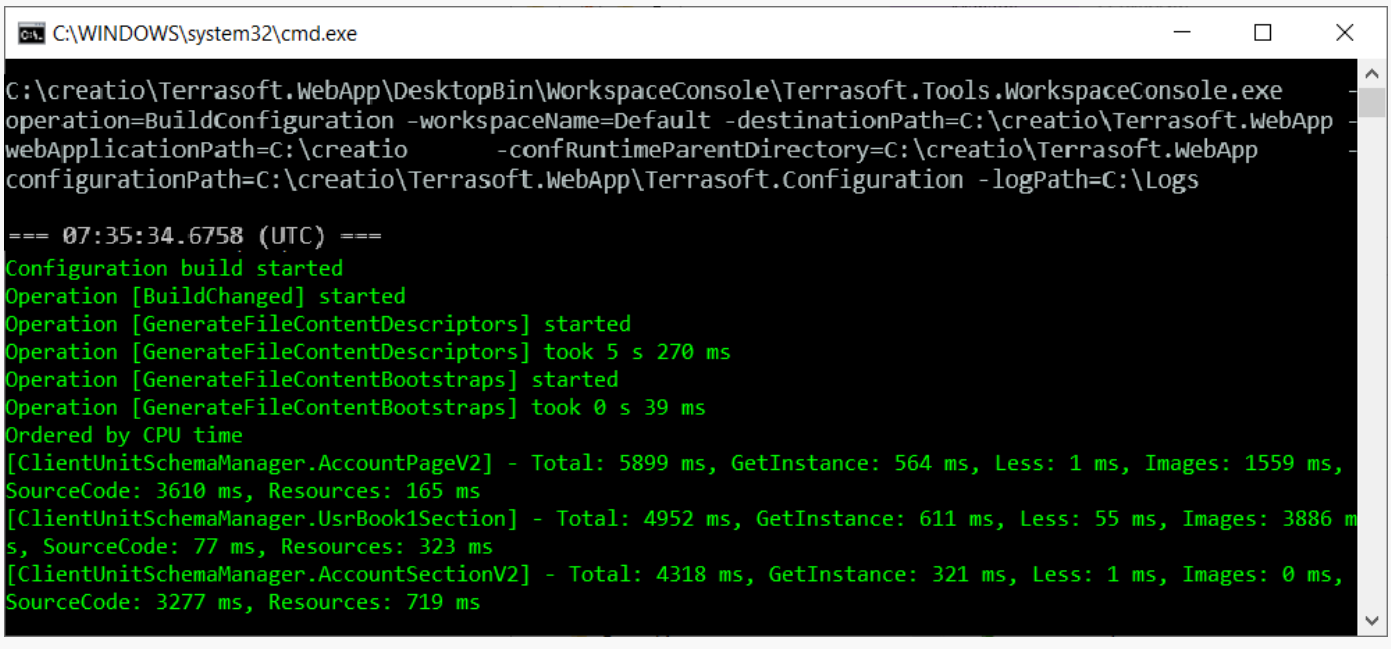
```
C:\creatio\Terrasoft.WebApp\DesktopBin\WorkspaceConsole\Terrasoft.Tools.WorkspaceConsole.exe
pause
```

Save the batch file.

4. Generate the static content in the file system

To **generate the static content in the file system**, double-click the batch file name.

This will open the console that will output the execution process of the operation specified in the corresponding WorkspaceConsole command.



As a result, the utility will generate the static content of the updated schemas in the file system.

WorkspaceConsole utility parameters CLI

Medium

`-help`

Displays the comprehensive list of utility parameters and their brief descriptions. If you specify other parameters, the utility ignores them.

`-operation`

The name of the operation to execute. Required. The default value is `LoadLicResponse`.

Possible values

`LoadLicResponse`	Uploads licenses to the database specified in the connection string. The only operation that does not require the `-workspaceName` parameter.
`SaveRepositoryContent`	Saves the *.zip package archives specified in the `-contentTypes` parameter from the directory specified in the `-sourcePath` parameter to the directory specified in the `-destinationPath` parameter.
`SaveDBContent`	Saves the database content to the file system. The `-contentTypes` parameter determines the content type. The `-destinationPath` parameter determines the content export path in the file system. Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`SaveVersionSvnContent`	Dumps the package hierarchy as a set of *.zip archives. The `-destinationPath` parameter determines the content export path in the file system. The `-sourcePath` parameter specifies the SVN repositories.
`RegenerateSchemaSources`	Re-generates the source code and compiles it. Requires the `-confRuntimeParentDirectory` parameter.
`InstallFromRepository`	Imports the package content and metadata from *.zip archives into the configuration. If needed, the utility runs bound SQL scripts, re-generates source code, and installs bound data. Works exclusively with updated or new

	packages and their elements. Requires either the `-webApplicationPath` or `-configurationPath` parameter. Also requires the `-confRuntimeParentDirectory` parameter.
`InstallBundlePackages`	Installs the bundle of packages listed in the `-packageName` comma-separated list to the workspace specified in the `-workspaceName` parameter. Requires either the `-webApplicationPath` or `-configurationPath` parameter. Also requires the `-confRuntimeParentDirectory` parameter.
`PrevalidateInstallFromRepository`	Checks if installing packages from *.zip archives is possible.
`ConcatRepositories`	Merges several repositories.
`ConcatSVNRepositories`	Merges several SVN repositories.
`ExecuteProcess`	Runs the business process in the configuration if the utility detects such a process. Requires the `-confRuntimeParentDirectory` parameter.
`UpdatePackages`	Updates the packages (`-packageName` parameter) in the database. Will only update the packages included in the product package hierarchy (`-productPackageName` parameter). Requires either the `-webApplicationPath` or `-configurationPath` parameter. Also requires the `-confRuntimeParentDirectory` parameter.
`BuildWorkspace`	Compiles the workspace (configuration). Use when developing schemas in Visual Studio. Requires the `-confRuntimeParentDirectory` parameter.
`ReBuildWorkspace`	Recompiles the workspace (configuration). Use when developing schemas in Visual Studio. Requires the `-confRuntimeParentDirectory` parameter.
`UpdateWorkspaceSolution`	Updates the solution and files of the Visual Studio project.
`BuildConfiguration`	Generates the static content in the file system. Uses the following parameters: • `-workspaceName` • `-destinationPath`

	- `-webApplicationPath` - `-logPath` - `-force` - `-configurationPath` Requires either the `-webApplicationPath` or `-configurationPath` parameter.
`RestoreConfiguration`	Restores the configuration from the package backup. Requires the `BackupConfiguration` operation to be performed. Requires the `-installPackageData` and `-backupPath` parameters.
`UpdateConfiguration`	Updates the configuration. Available in Creatio 8.0.4 and later.
`DeletePackages`	Deletes the packages from configuration. Available in Creatio 8.0.4 and later. Requires the `-packagesToDelete` parameter.
`ChangeFeatureState`	Changes the feature status. Available in Creatio 8.0.4 and later. Requires the `-featureCode`, `-featureState`, `-adminUnitName` parameters.

Attention. Execute the `BuildConfiguration` operation after executing the `InstallFromRepository`, `InstallBundlePackages`, `UpdatePackages` operations to ensure Creatio works correctly.

`-user`

Username for authorization. Specify the value if the utility's configuration file lacks a username or you need to execute the operation on behalf of a different user.

`-password`

User password for authorization. Specify the value if the utility's configuration file lacks a user password or you need to execute the operation on behalf of a different user.

`-featureCode`

The code of the additional feature. If you change the feature absent from the database, the utility adds the

corresponding record to the database table automatically. Required since version 8.0.4 for `ChangeFeatureState` operation.

`-featureState`

The status of the additional feature (`true` — turned on, `false` — turned off). Required for `ChangeFeatureState` operation since version 8.0.4.

`-adminUnitName`

The name of the user (the `[Name]` column value in the `[SysAdminUnit]` database table) that changes the feature status. Required for `ChangeFeatureState` operation since version 8.0.4.

If you specify the parameter, the utility modifies the corresponding record in the `[AdminUnitFeatureState]` database table.

If you omit the parameter, the utility sets the default value for the feature (the `[DefaultState]` column value in the [Feature] database table).

`-workspaceName`

Name of the workspace (configuration) where the exported packages are specified. By default, all users work in the `Default` workspace.

`-packagesToDelete`

The name of packages to delete. Can take multiple values separated by commas. Specify whether to delete the package as part of the configuration update. Required for `DeletePackages` operation and optional for `UpdateConfiguration` operation since version 8.0.4.

`-autoExit`

Specifies whether to end the utility process automatically after the operation finishes. Can be `true` or `false`. The default value is `false`.

`-processName`

Name of the process to run.

`-repositoryUri`

Address of the SVN repository that stores the package structure and metadata. Optional. If you specify the parameter, the utility uses its value instead of the value of the `-workspaceName` parameter.

`-sourceControlLogin`

Login of the SVN repository user.

`-sourceControlPassword`

Password of the SVN repository user.

`-workingCopyPath`

Local directory of the working copies of packages stored in SVN.

`-contentTypes`

Type of content exported to the drive from the database.

Possible values

`SystemData`	JSON data of system schemas. Dumps all system schemas and their columns except for data specified in the `-excludedSchemas` parameter.
`ConfigurationData`	JSON data of configuration element schemas. Dumps all schemas except for data specified in the `-excludedSchemas` parameter.
`Resources`	Localizable XML resources of configuration element schemas.
`LocalizableData`	Localizable XML content of configuration element schemas. Dumps only the text columns. Specify additional restrictions in the `-excludedSchemas` and `-excludedSchemaColumns` parameters.
`Repository`	ZIP workspace data.
`SqlScripts`	SQL scripts bound to packages.
`Data`	JSON data of system schemas and configuration element schemas. The combination of `SystemData` and `ConfigurationData` values.
`LocalizableSchemaData`	Data of localizable objects.
`All`	All content types.

`-sourcePath`

Path to a local directory from which to fetch the data. For example, packages, schemas, or resources. Can

accept several values separated by commas for the `ConcatRepositories` and `SaveVersionSvnContent` operations.

`-destinationPath`

Path to a temporary local directory to save the data. For example, packages, schemas, or resources.

`-webApplicationPath`

Path to the Creatio installation directory. The utility follows this path to fetch the database connection information from the `ConnectionStrings.config` file. If you omit this parameter, the utility connects to the database specified in the configuration string in the utility's configuration file.

The `-webApplicationPath` parameter specifies the path to the Terrasoft.WebApp directory for the `BuildWorkspace`, `ReBuildWorkspace`, and `UpdateWorkspaceSolution` operations.

`-configurationPath`

Path to the `..\Terrasoft.WebApp\Terrasoft.Configuration` directory. The utility exports the source code and resources of custom package schemas to this directory in the file system development mode.

`-backupConfiguration`

Specifies whether to back up the configuration before package installation. Available in Creatio version 8.0.2 and later. The default value is `true`. Optional for `InstallFromRepository` operation since version 8.0.2. Optional for `UpdateConfiguration` operation since version 8.0.4.

`-filename`

File name. Required for the `LoadLicResponse` operation.

`-excludedSchemas`

Names of configuration element schemas to exclude from the operation.

`-excludedSchemaColumns`

Names of the columns of configuration element schemas to exclude from the operation.

`-excludedWorkspaceNames`

Names of the workspaces (configurations) to exclude from the operation.

`-includedSchemas`

Names of configuration element schemas to forcibly include in the operation.

`-includedSchemaColumns`

Names of the columns of configuration element schemas to forcibly include in the operation.

`-cultureName`

Language culture code. Required if you specify the `Resources` and/or `LocalizableData` values in the `-contentTypes` parameter.

`-schemaManagerNames`

Names of configuration element schema managers in the operations. The default value is `EntitySchemaManager`.

`-packageName`

Package name in the configuration specified in the `-workspaceName` parameter. The utility downloads the packages on which the current package depends as well. If you omit the parameter, the utility downloads all packages of the configuration.

`-clearWorkspace`

Specifies whether to clear the workspace before the update. Can be `true` or `false`. The default value is `false`.

`-backupPath`

Path to a local directory that contains the *.gz archives of package backups.

`-installPackageSqlScript`

Specifies whether to run the SQL scripts before and after saving the packages. Can be `true` or `false`. The default value is `true`.

`-installPackageData`

Specifies whether to install the bound package data after saving the packages. Can be `true` or `false`. The default value is `true`.

`-updateDBStructure`

Specifies whether to update the database structure after saving the packages. Can be `true` or `false`. The default value is `true`.

`-regenerateSchemaSources`

Specifies whether to re-generate the source code after saving the packages to the database. Can be `true` or `false`. The default value is `true`.

`-continueIfError`

Specifies whether to abort the installation after receiving the first error. If set to `true`, the installation process goes through to the end. You will receive the list of the errors encountered. Can be `true` or `false`. The default value is `false`.

Always set the `-continueIfError` parameter to `false` for the `InstallFromSvn` and `InstallFromRepository` operations. These operations manage updated or new packages and their elements. The utility detects the updated elements by comparing the new and existing package structures. As such, if you run the command without specifying the `-continueIfError=true` key and an error occurs, then rerun the command for the same configuration, the second command will finish without errors yet will not update the database. In this case, the first operation synchronizes the package structures of the configuration and repository, meaning the second operation will be unable to detect the updated package elements.

`-skipCompile`

Specifies whether to run the compilation. Will work if you set the `-updateDBStructure` parameter to `false`. Can be `true` or `false`. The default value is `false`.

`-autoUpdateConfigurationVersion`

Updates the configuration version in the database to the Creatio version. Can be `true` or `false`. The default value is `false`.

`-warningsOnly`

Specifies whether to display an error notification if the utility detects an operation execution error. Can be `true` or `false`. The default value is `false`.

`-confRuntimeParentDirectory`

Path to the parent directory of the `..\Terrasoft.WebApp\conf` directory. You must use the parameter in commands that compile Creatio or generate the static content:

- `RegenerateSchemaSources`
- `InstallFromRepository`
- `InstallBundlePackages`
- `UpdatePackages`

- `BuildWorkspace`
- `RebuildWorkspace`
- `ExecuteProcess`