Kentico provides a continuous integration solution that allows you to serialize the data of objects from the database into XML files on the file system. You can then add the files to a source control system (for example Team Foundation Server or Git) and use them to synchronize database data between team members. The system ensures that the XML data of matching objects is always identical and consistent (including element and attribute order), even when serialized on different instances of Kentico.

The system stores the XML files containing the serialized data of objects in the project's **CMS\App_Data\CIRepository** folder. For details about the folder and file structure, see Continuous integration repository structure.

If you need to work with the continuous integration files in a different location, use one of the approaches described in Storing the continuous integration files in a custom location.

Continuous integration overview

The continuous integration solution is usable in the following environment:

- Multiple development instances, each with its own project files and database
  – AND –
- A central source control system that provides change management and version control for files

To start developing sites using the continuous integration solution:

1. Set up your development environment and continuous integration according to the instructions in:
   - Preparing the development environment
   - Excluding objects from continuous integration
2. Establish a process for transferring objects back into the database from the *CIRepository* XML files. See Restoring continuous integration files to the database and Synchronizing database changes not managed by continuous integration.
3. Begin development. We strongly recommend following the best practices described in Using continuous integration with Visual Studio. The page primarily focuses on using Visual Studio in combination with a *Team Foundation Version Control* or *Git* source control repository, but many of the recommendations also apply generally to any source control system.

Usage and relationship with other features

Continuous integration is designed to allow synchronization and version control of database data ***during development*** (through an external source control system).

Continuous integration has a ***negative impact on site performance***. The impact is not significant for development instances with a low number of users, but continuous integration should not be enabled on production sites. For more information about deployment options, see the Deploying data to production sites section.

Additionally, we do not recommend using continuous integration together with the other team development and versioning features in Kentico:

- Object locking
- Object versioning (including the Recycle bin for objects and Pages)
- External editing of code (and Deployment mode for virtual objects)

Continuous integration always works with the main version of each object in the database. Older or deleted versions of objects and non-published versions of pages (including their child objects) are ignored. For example:

- If you use the object versioning feature, continuous integration ignores all versions of objects except for the latest. If you roll back an object to an older version, the object is updated in the *CIRepository* folder.
- If you delete an object or page to the recycle bin, continuous integration fully removes the corresponding XML file. If you restore the object, an XML file is added as if a completely new object were created.
- If you edit an object's code externally (e.g. the code of a CSS stylesheet), changes are ignored by continuous integration until you synchronize the code back into the database. You can still synchronize the code without the continuous integration solution by including the external file in your source control.

Page workflow

When using the page workflow feature, the majority of workflow information is not tracked by continuous integration. This includes moving pages to different workflow steps and using the check-in/check-out functionality. Pages under workflow are fully stored in the repository only after they are published. When creating a new page, the page data is stored without its child objects. If you make changes to the content of a page, non-versioned page data is stored immediately while versioned page data is stored only when the page is published.

> **Note**: The workflow functionality itself (including Content locking and Page versioning) is not affected when used together with continuous integration. For example, your content editors can use basic and advanced workflows to manage life cycles of pages on a single Kentico instance. When pages reach the *Published* step, the page content is stored in the repository.

Limitations

Continuous integration currently has the following limitations:

- Continuous integration only tracks object changes made through the Kentico administration interface or API. Changes made directly in the database or by external tools that modify the database are not tracked. After making such changes, you need to manually serialize all objects in the *Continuous integration* application to synchronize the database with the file system repository. For example, continuous integration does not track roles imported by the AD import external utility.
- You ***cannot*** reliably use continuous integration in environments where the *CIRepository* folder is located on a ***shared file system*** (multiple Kentico instances with separate databases connected to the same file system repository). Common examples of shared file systems are external storage providers, such as Azure storage or Amazon S3.
- Object types that represent live site data or are strictly related to specific instances are not supported. See Object types supported by continuous integration for a complete list of supported object types.

Preparing the development environment

Before you can use the continuous integration solution, you need to set up Kentico instances for your development team.

Start by preparing an initial instance:

1. Either install a new instance of Kentico or select an existing instance as a starting point.
2. Enable continuous integration on the given instance (see Enabling continuous integration for details).
3. Create a backup of the instance's database.
4. Add the ***entire*** Kentico solution to your source control (adding only the *CIRepository* folder may lead to problems and is not recommended).

To add development instances into your environment, perform the following steps for each instance:

1. Connect to the source control from the development machine and load the latest version of the solution files onto the local file system.
    - Decide how to handle source control for the project's **web.config** file. Each development machine needs to have a connection string to a different database, but you may want to synchronize other parts of the web.config.

      > See https://weblog.west-wind.com/posts/2013/Feb/27/Sql-Connection-Strings-in-Config-Files-vs-Source-Control for possible solutions.

2. If you use IIS to run your development sites, register the project as an application in IIS:
    a. Open *Internet Information Services (IIS) Manager*.
    b. Add a new application under your IIS web site and map the *Physical path* to the project's *CMS* folder.
    c. Create a new application pool for the application (or use an existing application pool).
3. Create (restore) a copy of the initial database on your SQL server.
4. Connect the instance to the new copy of the database (set the appropriate connection string in the project's web.config).

After you complete the process, all development machines will contain their own Kentico project files connected to a separate copy of the database. You can start using your source control system to synchronize project files between the instances. The continuous integration solution serializes database data onto the file system and you can include it in your source control, just like any other files.

> The process described above ensures that objects have identical GUID values (globally unique identifiers) across your entire development environment. If you perform a separate database installation for each development instance, you will need to manually resolve a large number of GUID conflicts when synchronizing data through the continuous integration solution.

If you need to add further instances later (after development with continuous integration starts), use the same process. However, you need to make sure the database is synchronized with the rest of the development environment. Use one of the following approaches:

- Maintain a central database connected to the mainline source control and always create an up-to-date backup for new developers.
  – OR –
- Use the original database backup and then get the latest data:
  1. Restore the current object status from the continuous integration repository into the database. See Restoring continuous integration files to the database.
  2. Manually transfer changes made to the data of object types not supported by continuous integration (maintain documentation of such changes). You can use the export and import feature.

Enabling continuous integration

To configure a Kentico instance to use the continuous integration solution, perform the following steps:

1. Disable running of scheduled tasks (using the **Settings -> System -> Scheduled tasks enabled** setting).
2. Open the **Continuous integration** application in the Kentico administration interface.
3. Select the **Enable continuous integration** checkbox.
4. Click **Save**.
5. Click **Serialize all objects**.
6. Wait until the serialization process finishes and then re-enable scheduled tasks.

The project's **CMS\App_Data\CIRepository** folder now contains XML files storing the serialized data of all supported objects from the database. The system also tracks create, update and delete operations for the given objects and automatically transfers the changes to the serialized data on the file system.

Configuring instances to synchronize macros

> For detailed information, see: Working with macro signatures

To ensure that macro expressions work correctly when synchronizing objects using the continuous integration solution, all development instances must use the same hash salt value.

If your development instances do not all start with the same web.config file, you need to set a matching value for the **CMSHashStringSalt** key in the *appSettings* section on all instances. The best option is the hash salt used by the starting instance that you used to create your initial database backup. You can use any string as the value, but the salt should be random and at least 16 characters long. For example, a randomly generated GUID is a strong salt:

```
<add key="CMSHashStringSalt" value="e68b9ad6-a461-4707-8e3e-ece73f03dd02" />
```

The salt value is used as part of the input for the hash function that creates the security signatures of macros. Having the same hash salt value on all instances is necessary to ensure that macro signatures are valid when transferring data between instances.

The best option is to set the hash salt value before you start development. Changing the salt causes all current hash values to become invalid. To fix existing macro expressions in the system after changing the hash salt, you need to re-sign the macros.

Synchronizing macros between instances with different users

You may also encounter problems with invalid macros if you do not synchronize all users between the instances in your environment. Macros are not valid if the user in the signature does not exist on the given instance.

To ensure that all macros work correctly regardless of the available users, set up **macro signature identities**:

1. Find groups of users in your environment who require the same permissions.
2. In the **System** application, create a macro identity object for each group on all instances, with a matching **Identity name**.
3. Assign an **Effective user** with appropriate permissions to each macro identity. The effective user can be different on each instance.
4. Assign the macro identity to the appropriate users in the **Users** application.

Macros will now be signed using the assigned identities instead of user names. The identities are available on all instances, so you do not need to synchronize the user accounts.

Deploying data to production sites

Deployment of content and data to instances that host live websites is not the primary purpose of the continuous integration solution. Instead, we recommend using the Staging functionality for deployment:

1. Set up one of the instances in your continuous integration environment as a source server for staging.
2. After restoring continuous integration data on the given instance, synchronize the changes via staging tasks to any required target servers.

If you use staging to deploy pages, we also recommend setting the **CMSStagingUseAutomaticOrdering** key to *false* in the *appSettings* section of the *web.config* file on all ***target servers***.

```
<add key="CMSStagingUseAutomaticOrdering" value="false" />
```

The key ensures that the order of pages in the content tree remains consistent after staging. If you leave automatic ordering enabled on the target staging servers, the **Content -> Content management -> New page ordering** setting of each site may override the original page order created by the continuous integration restore process.

Deploying via continuous integration

If you do decide to deploy data to a live production site via the continuous integration restore functionality, use the following process:

Warning

Restoring data through continuous integration ***deletes*** any pages and supported objects that are missing in the *CIRepository* folder. You may lose content created directly on the production instance through standard editing or live site interaction.

For example, the following scenario is possible:

- You prepare a content update in your development environment and copy the *CIRepository* folder to the production instance.
- Live site users meanwhile create new pages via user contributions.
- You restore the *CIRepository* content to the database.
- The pages added by the live site users are deleted, because they are not included in the restored data.

Optionally, you can configure the repository.config file on the production instance to exclude any object types that you do not wish to update and overwrite on the live site.

1. Leave continuous integration disabled on your production instance of Kentico (the setting in the *Continuous integration* application). Continuous integration has a ***negative impact on site performance*** and should not be enabled on production sites.

   Important: You can restore data even when continuous integration is disabled.

2. Upload the *CIRepository* folder containing the required data into the *CMS\App_Data* folder of your production instance.
3. Restore the *CIRepository* content to the production database – see Restoring continuous integration files to the database.
   - We strongly recommend performing the restore process at a time when the site has minimal traffic and no editing is taking place.
   - The restore process may take longer than usual when continuous integration is disabled.
4. Restart the application (if not done automatically by your restore process). Open the *System* application and click *Restart application* on the *General* tab.

The data is now deployed to the production database. Keep in mind that any changes made in the administration interface or on the live site will not be tracked and reflected in the *CIRepository* folder.

Upgrading projects that use continuous integration

If you need to apply a hotfix or upgrade to an instance that has continuous integration enabled, use the following procedure:

1. Restore objects from the continuous integration repository to your database.

   You can skip the restore if you are sure that your database is synchronized with the current status of your *CIRepository* folder.

2. Apply the hotfix or upgrade.
3. Run complete serialization for all objects to recreate the content of the *CIRepository* folder:
   a. Disable running of scheduled tasks (using the **Settings -> System -> Scheduled tasks enabled** setting).
   b. Open the **Continuous integration** application in the Kentico administration interface.
   c. Click **Serialize all objects**.
   d. Wait until the serialization process finishes and then re-enable scheduled tasks.

This approach ensures that your *CIRepository* folder contains all object changes made by the hotfix or upgrade and that the serialization process itself runs according to the new version.

For hotfixes, you need to apply the update separately for each development instance. After one developer commits the hotfixed changes to the source control, other developers CANNOT commit or load changes until they apply the hotfix to their own instance.

For major upgrades, it may be more efficient to upgrade only one instance and then set up your environment again according to the process described in Preparing the development environment.

Storing the continuous integration files in a custom location

After you set up and enable continuous integration, the system serializes database data into XML and stores the results on the file system.

By default, the files are created in the Kentico project's **CMS\App_Data\CIRepository** folder. If your development environment requires a different location for the continuous integration files, you can use one of the following approaches:

- Create a file system junction point
  – OR –
- Use a custom Kentico storage provider

> **Important**
>
> We recommend keeping the default *CIRepository* folder within the *App_Data* folder of the Kentico web project and having the entire Kentico solution included within your source control. This setup allows you to easily synchronize all types of related project files together with the database data tracked by the continuous integration solution.
>
> If you do decide to change the *CIRepository* location, do NOT use the new folder as the root of your source control repository. Always add it as a subfolder under a different root folder. The continuous integration solution **deletes all folders** within *CIRepository* when storing all objects. If the folder is the root of your source control repository, you may encounter errors or other problems when using source control systems that utilize custom folders in the root.

Creating a junction point

By defining a file system Junction point, you can redirect the *CIRepository* folder to a different location.

> **Limitation**: The target location must be on a local drive or volume (network locations cannot be used).

1. If you have already enabled continuous integration and serialized objects, you need to:
   a. (Optional) Back up your *repository.config* and the data files in *CIRepository*.
   b. Delete the *CIRepository* folder in your Kentico project (it is not possible to create a junction point for an existing folder).
2. Create the target folder where you wish to store the continuous integration files.
3. Open the Windows command prompt and run the **mklink /J** command:

```
mklink /J <CIRepository folder path> <target folder path>

For example:
mklink /J C:\inetpub\wwwroot\Kentico\CMS\App_Data\CIRepository C:
\ExternalSourceControl\CIRepository
```

When serializing object data to the file system, the Kentico continuous integration solution now creates the files in the specified target folder. You can add the target folder into your source control system.

Using a custom storage provider

You can use a custom storage provider to change the path of the *CIRepository* folder. The target location can be a local drive, or any network location for which the application has sufficient permissions.

Limitations

- The target location always includes the full *~/App_Data/CIRepository* folder structure, starting from a specified root directory.
- After you define the custom storage provider, the *CIRepository* folder can no longer be accessed within the original location (unlike when using a junction point).

To register a storage provider, you need to run custom code during the initialization of the application. For example, the following steps demonstrate how to change the *CIRepository* path using a custom module:

Important: The code must be added into a custom project within the Kentico solution. Customizations defined directly within the Kentico web project (e.g. in the *App_Code* folder) cannot be detected by the external utility used to restore continuous integration data.

1. Open the Kentico web project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a new *Class Library* project (assembly) in the solution or reuse an existing custom project.
3. Add references to the required Kentico libraries (DLLs) for the project:
   - Right-click the project and select **Add -> Reference**.
   - Select the **Browse** tab of the **Reference manager** dialog, click **Browse** and navigate to the ***Lib*** folder of your Kentico web project.
   - Add references to the following libraries:
     - **CMS.Base.dll**
     - **CMS.Core.dll**
     - **CMS.DataEngine.dll**
     - **CMS.IO.dll**
4. Add a reference from the Kentico web project *(CMSApp* or *CMS)* to the custom project.
5. Edit the project's **AssemblyInfo.cs** file (in the *Properties* folder).
6. Add the **AssemblyDiscoverable** assembly attribute:

```
using CMS;

[assembly:AssemblyDiscoverable]
```

7. Create a new module class under the custom project.
8. Override the module's **OnInit** method and map the *CIRepository* folder to a custom location.

Example

```
using CMS;
using CMS.DataEngine;
using CMS.IO;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(CustomCIStorageModule))]

public class CustomCIStorageModule : Module
{
        // Module class constructor, the system registers the module under the
name "CustomCIStorage"
        public CustomCIStorageModule()
                : base("CustomCIStorage")
        {
        }

        // Contains initialization code that is executed when the application
starts
        protected override void OnInit()
        {
                base.OnInit();

                // Maps the 'App_Data/CIRepository' folder in the Kentico project
to a custom location
                // The 'UseLocalFileSystemForPath' method internally creates a
custom StorageProvider instance,
                // and maps the specified Kentico directory (first parameter) to
a different location based on a root path (second parameter)
                StorageHelper.UseLocalFileSystemForPath("~/App_Data
/CIRepository", @"C:\ExternalSourceControl");
        }
}
```

9. Save all changes and rebuild the Kentico solution.

When serializing object data to the file system, the Kentico continuous integration solution now uses the *CIRepository* folder located under the mapped root folder. You can add the folder into your source control system.

Configuring the character encoding for repository files

By default, the continuous integration solution generates non-binary files in the *CIRepository* folder using UTF-8 character encoding.

If you need to use a different encoding type for the repository files (for example due to requirements of a file comparison tool), add the **CMSCIEncoding** key to the *appSettings* section of your project's **web.config** file. For example:

```
<add key="CMSCIEncoding" value="utf-16" />
```

The *CMSCIEncoding* key supports values matching the encoding names listed in the Encoding Class article.

Note

Changing the value of the *CMSCIEncoding* key does not update the encoding type of existing files in the *CIRepository* folder. To fully update the encoding of the repository content, you need to:

- Run complete serialization for all objects.
- Manually update the encoding type of your *repository.config* file.