This documentation section assumes that you are familiar with the System.IO library and know how to use it to manipulate files and directories in the Windows file system. Learn how to use System.IO.

Working with CMS.IO is in most aspects the same as working with System.IO. We recommend using CMS.IO instead of System.IO in your code so that your custom functionality is not dependent on a single file system type.

Similarities between CMS.IO and System.IO

Classes

The following classes defined by CMS.IO are identical in function to their respective counterparts in the System.IO library:

- Directory
- DirectoryInfo
- File
- FileInfo
- FileStream
- Path
- StreamReader
- StreamWriter
- StringReader
- StringWriter

Enumerations

The following enumerations are also identical:

- FileAccess
- FileAttributes
- FileMode
- FileShare
- SearchOption

Other classes and enumerations

For classes and enumerations not implemented by CMS.IO, we recommend using the standard implementations from the *System.IO* namespace (for example System.IO.Stream). Kentico expects these System.IO types across its API.

Differences between CMS.IO and System.IO

The most significant difference is in the creation of new instances of objects. Instead of a constructor, each class contains a **New()** method, which accepts the same parameters as the class' constructor.

The following example shows how to write text into a file. You can see that the instance of the FileInfo class is created using the New() method. Please note that the **StreamWriter** class used in the example is also a member of CMS.IO, not System.IO.

```
using CMS.IO;

...

FileInfo fi = FileInfo.New("MyFile.txt");

using (StreamWriter sw = fi.CreateText())
{
    sw.WriteLine("Hello world!");
}
```

There are a number of types which can be found in System.IO, but are not implemented in CMS.IO. These include seldom used classes, class members and methods. Additionally, CMS.IO does not contain any definitions of exceptions. You need to use System.IO exceptions or implement your own.

Helper methods

CMS.IO contains a number of additional methods and properties, which simplify operations with files and directories. The following list describes the most useful methods:

DirectoryHelper class

- `void DeleteDirectoryStructure(string path)` - deletes the directory specified by the path parameter and all underlying directories.
- `void EnsureDiskPath(string path, string startingPath)` - checks whether all directories between **startingPath** and **path** exist and creates them if necessary.
- `void EnsurePathBackslash(string path)` - adds a backslash to the end of the path specified if the backslash is missing.

Directory class

- `void PrepareFilesForImport(string path)` - converts all physical media file names to lower case to ensure compatibility with case-sensitive file systems.

StorageHelper class

- `IsExternalStorage` - this property indicates if the application is using a storage other than the default Windows file system. Returns true if the **CMSExternalStorageName** web.config key is set. See Configuring file system providers for more information.