Configuring the File System

Product version:	4.41
Document version:	1.0
Document creation date:	01-02-2005

Purpose

This technical note describes how to configure and set up the file system, Unified File System.

Table of Contents

Configuring web.config

What is EPsUploadDir?

Without any configuration, a handler will by default be set up to match whatever configuration made to `EPsUploadDir` setting under the `appSettings` element in `web.config`.

After you add file system configuration, the `EPsUploadDir` setting will only act as the default folder opened in any graphical user interfaces in EPiServer. It will also act as the default folder for special page folders until you specify `pageDirectory="True"` on a handler. This is further described in the following chapters.

Reconstructing Default Configuration

By default, EPiServer has no configuration information about the file system setup in `web.config`. It will, by default, configure a backwards-compatible setup, based on the value in `EPsUploadDir` in `web.config`. Before you begin this configuration tour, you need to add this information inside the root configuration element of your `web.config` file.

1. Add the configuration setting to let ASP.NET know we have our own section.

```
<configSections>
    <sectionGroup name="episerver">
      <section name="unifiedFileSystem" allowDefinition="MachineToApplication"
allowLocation="false"
type="EPiServer.FileSystem.Handler.ConfigurationHandler,EPiServer" />
    </sectionGroup>
  </configSections>
```

   Important! The `configSections` tag must be located at the top of `web.config`. This does not apply to the `episerver` section.

2. Add the configuration section for the default upload directory.

```
<episerver>
    <unifiedFileSystem>
      <handlers>
        <handler pageDirectory="False" virtualName="upload"
virtualShare="False"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer" />
      </handlers>
```

```
    </unifiedFileSystem>
  </episerver>
```

3. After you save your changes, your installation should behave as before, because you have not change any behaviour.

Note The virtual name `upload` above can, of course, be set to another location in your installation, just check the value of `EPsUploadDir`.

Adding New Folders

We want the editors to be able to access the `images` directory, which defines many images used in the Web site design.

- Add a new handler section which points out the local images folder.

```
<handler pageDirectory="False" virtualName="images" virtualShare="False"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer" />
```

The handler section is identical, but we change the `virtualName` to `images`, which is the name of our folder. If you open the file manager in Edit mode, you should have two directories pointing to both the images folder and the upload folder.

Important! You must set the correct Windows ACL on folders exposed to editors to make sure they have enough permissions to edit files and folders. On Windows 2000/XP, you should give the local ASPNET account "Change" permissions on the folder. On Windows Server 2003, use the IIS_WPG group or the actual process account configured in Internet Information Services (IIS).

Adding New Secure Folders

We have until now been adding handlers for folders that already exist on the Web site. In reality we make sure EPiServer can display them in Edit mode. When a user downloads an image from the `images` folder in the previous example, it will be served the Internet Information Services, just as we are used to.

If we want to add a folder that will be secured by EPiServer, the scenario is different.

1. Make sure the folder cannot be accessed by IIS, as this would short-circuit the security check.
2. Add a new folder to `c:\inetpub` and call it `SecureFiles`.
3. Add a handler configuration section.

```
  <handler pageDirectory="False" virtualName="Secure" virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="c:\inetpub\SecureFiles" />
        </handler>
```

When you open the file manager in Edit mode, you will not see the new folder, because no one has been granted access yet. Go to Admin mode and click *File management* under the *Tools* section. When you select the Secure folder, you will see a new button, *Security*. Add the appropiete access levels.

The main difference from the previous configuration is that we set `virtualShare` to `True`. This will tell EPiServer that files access in the `Secure` folder must pass security checks, both when accessing it from Edit mode and when downloading files.

Files that don't get exposed by Internet Information Services can be downloaded anyway, because the technology is the same as that for simple address to pages in EPiServer. We add a custom Web form to handle traffic that results in "404 File Not Found from IIS". So if you access

http://localhost/Secure/MyFile.doc, it will be delivered to the user by EPiServer instead, and the user will not be able to see any difference. This makes it easy to take an existing directory and move it to a secure location as described in the next example without changing any links.

Securing Existing Folders

Securing an existing folder is actually not very different from the previous example. We assume that we have an `Extranet` folder on our Web site that contains documents we wish to secure.

1. Set up the configuration section in `web.config`.

```
 <handler pageDirectory="False" virtualName="Extranet" virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="c:\inetpub\Extranet" />
        </handler>
```

2. Take the existing `Extranet` folder and move it from `c:\inetpub\EPiServer\Extranet` to `c:\inetpub\Extranet`. This means that we take it out of Web exposure as far as IIS is concerned.
3. Go to Admin mode and click *File management* under the *Tools* section. When you select the Extranet folder, you will get a new button, *Security*. Add the appropriate access levels.

You should now be able to access the files just as before, but every file is checked for security by EPiServer, as defined in Admin mode.

Moving Special Page Folders to a Separate Folder

EPiServer has a concept of special page folders, where every page can have its own folder with files that only apply to this specific page. These folders have previously been stored in the same upload folder as the rest of the files. For example `/upload/33/MyFile.doc` where ”33” is a folder created specifically for a page in EPiServer and filtered out of the other views of the upload folder.

Moving these folders to another location is very simple and the `pageDirectory` attribute that has been set to False in the previous examples is, of course, the key to success.

1. Create a new folder called `pages` on your Web site and add a configuration section for this folder.

```
<handler pageDirectory="True" virtualName="pages" virtualShare="False"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer" />
```

2. If you browse the file manager in Edit mode, you will not see the `pages`, folder, but if you click the `Current page` folder, you will be linked to a special folder for this page that resides in the `pages` folder.

Note If you are adding this configuration to an existing site, you have to move the actual numeric folders from the `upload` folder to the `pages` folder.

Securing Special Page Folders

Sometimes you have Web sites with lots of different access levels or may need to make sure that documents in the special page folder do not become accessible before the page. As seen in other configuration examples, we can secure page folders in the same way.

1. Move the folder created in the previous example, or create a new one at `c:\inetpub\pages`.
2. Change the page configuration (you cannot have more than one page folder of course).

```
<handler pageDirectory="True" virtualName="pages" virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
              <customSettings PhysicalPath="c:\inetpub\Pages" />
         </handler>
```

You cannot browse page directories; you will only be able to access them directly in the file manager using the `Current page` folder.

Note If you access files in a page folder, they will receive the same security checks as a page would. For example, a page were the start publish date is forward in time will require "Edit" access to files in this folder, just as for the page itself.

Moving Directories Around / Mappings

When presented with this amount of configuration settings, you will probably find yourself in a situation where you need to move a directory from one location to another.

The functionality presented below was built to target existing EPiServers that want to enable security on some folders on their site. If you move existing secured directories you should consider the previous security notices.

For example, if you have files in `/upload/Documents` that you wish to move to the previously created folder `Secure`, all existing links would point to the wrong location. One solution is to find and change all links, so that they point to the new location. Another way is to use the mapping features in the file system, for example:

```
<mappings>
 <map fromPath="/upload/documents" toPath="/Secure"
type="EPiServer.FileSystem.Mapping.SimpleMapping,EPiServer" />
</mappings>
```

This configuration will ensure that access to `/upload/documents/MyFile.doc` will be redirected to `/Secure/MyFile.doc`. Mappings only work when the `toPath` is a `virtualShare.` Otherwise you should use IIS to redirect requests.

A more complex operation would be to only move the special page folders from the `upload` folder to the `pages` folder without changing the existing links. The example configuration requires that you have set up a secured `pages` folder in the previous example.

```
<mappings>
   <map fromPath="/upload" toPath="/pages"
type="EPiServer.FileSystem.Mapping.PageDirectoryMapping,EPiServer" />
</mappings>
```

As you can see, we now use the `PageDirectoryMapping` type instead. The mapper will only redirect requests that start with `/upload`, followed by one or more digits. For example, `/upload/45/Myfile.doc` will be redirected to `/pages/45/MyFile.doc`.

Configuration Search Capabilities

Mapping to an Indexing Service Catalog

In the file manager in Edit mode, you have an option to search in the file system. This feature uses Microsoft Indexing Service. By default, this feature will search in the SYSTEM catalog, which by default in Windows allows search in all local drives. Sometimes this catalog is not available, maybe

deleted for security reasons, so you have to map a new catalog. This is done by the following configuration in EPiServer:

```
<handler pageDirectory="False" virtualName="Extranet" virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings IndexCatalog="ExtranetCatalog"
PhysicalPath="c:\inetpub\Extranet" />
        </handler>
```

The catalog is created in Indexing Service and called `ExtranetCatalog`. It maps to a directory either identical to `c:\inetpub\Extranet` or any folder containing the Extranet folder. The file manager will automatically set scope for the search to the correct path, so the path of the folder does not have to match the path of the catalog. Remember that if you have more that one handler, you must configure `IndexCatalog` for all of them, for example by using the same Index Server catalog.

This will make sure you can search for files in Edit mode, but if you want the search template in EPiServer to search the same way, you have to make some changes to this template as described in the next step.

Secured Folder Support for Pre-4.41 Templates or Custom Templates

File Listing Template

Note This section only applies if you use the file listing template.

The file listing template does not support Unified File System. It was built to support Windows files and folders and must remain backwards-compatible. The core of the file listing template is the `FileTree` control that must be changed to `UnifiedFileTree`, which has very similar syntax, but with some minor changes. The `UnifiedFileTree` only supports directories defined in Unified File System and should be considered for other reasons such as security concerns, as it will not let an editor set the path to a directory not defined in Unified File System.

You need to recompile the project for this change. Please see `Templates/Units/FileListing.ascx` in the EPiServer 4.41 templates pack for example code.

Search Template

If you want the search template in EPiServer to search through Unified File System instead to make sure the results reflect the user access rights, you must make a small code change.

This is the default code for the search template in `Templates/Units/Search.ascx`. The settings `MainScope` and `MainCatalog` must be removed to transfer control to Unified File System:

```
<episerver:PageSearch
               Runat="server"
               ID="SearchResults"
               SearchQuery='<%# SearchQuery.Text %>'
               SearchFiles='<%# SearchFiles.Checked %>'
               OnlyWholeWords='<%# OnlyWholeWords.Checked %>'
               MainScope='<%# CurrentPage["MainScope"] %>'
               MainCatalog='<%# CurrentPage["MainCatalog"] %>'
               PageLink='<%# Configuration.StartPage %>'
               PageLinkProperty="MainContainer"
>
```

Add the setting `UnifiedSearchLocations` instead which is a comma-separated list of directories to search in:

```
<episerver:PageSearch
               Runat="server"
               ID="SearchResults"
               SearchQuery='<%# SearchQuery.Text %>'
               SearchFiles='<%# SearchFiles.Checked %>'
               OnlyWholeWords='<%# OnlyWholeWords.Checked %>'
               UnifiedSearchLocations='/upload,/Extranet'
               PageLink='<%# Configuration.StartPage %>'
               PageLinkProperty="MainContainer"
>
```

Custom Templates

If you have other templates in your site that programmatically access files using Microsoft.NET API to any of the directories that will be secured, you must change this, so that they use the Unified File System API instead. The Unified File System API is very similar to the Microsoft.NET API so there should not be any problems migrating code. Code that only accesses non-secured directories does not need to be modified. The recommendation is to always use our API as it is specifically designed for the Web and EPiServer.

Example:

```
//Example using Microsoft.NET API
FileInfo file = new FileInfo( Server.MapPath("~/securefolder") );
Response.Write(file.Length.ToString());
```

```
//Example using Unified File System API
UnifiedFile file = UnifiedFileSystem.GetFile("/securefolder");
Response.Write(file.Length.ToString());
```

Security Check List

- Your secured directory should not be exposed by Internet Information Server in any way. It would bypass Unified File System security checks, which are responsible for downloads.
- The search template should not use Index Server directly. It is the responsibility of Unified File System to handle Index Server communication.
- Do not expose the same directory more than once. For example, if you were to expose some secure documents through both `/Files/Extranet` and `/Extranet`, they would get different access control lists, based on the URL.
- Do not rename directory handler names without reconfiguring security settings, for example changing `/Extranet` to `/AnotherExtranet` would make it loose security settings.

Technical Details and Troubleshooting

Access Rights are Mapped to the URL

When you add a folder and set access rights, these settings are stored in EPiServer with the URL to access the folder, not the physical path. For example, if you set that `/Secure/Documents` should have some specific access rights, you could change the physical location of the actual folder without affecting access rights set in EPiServer, as long as it is accessed using the same URL. This makes it easy to move a site or the locations of folders in the file system. This also means that EPiServer never touches the file system to store any internal settings. All settings made in EPiServer are stored in EPiServer.

Security notice 1 Never change the names of subfolders in Windows Explorer, as security settings will be lost. This does not apply if a subfolder is set to inherit security settings from the parent folder. If you must allow editors to change subfolders in Windows Explorer, you should consider only setting access rights to the root folder, as the name of this folder is stored in `web.config`.

Security notice 2 EPiServer controls access rights using the URL to the files, so you must never expose the same folders using different virtual names. For example, if you were to expose `C:\Secure` both as `/Secure` and as `/Public`, you will allow visitors to access the same files using different paths and access rights, if you haven't set the identical access rights in EPiServer of course.

I Get a NullReferenceException in File Manager

Check that `EPsUploadDir` points to an existing directory.

I Configured "/Upload" to Be the Page Directory, but I Can't See it Anymore

The `pageDirectory` setting should only be used if you want a special directory separated from the other directories to store page folders. You will not be able to browse a page directory and it will only be used to store the special page folders. A page folder can be accessed using the “Current page” link in the file manager.

Is *Impersonation* Required/Recommended?

No, Unified File System moves responsibility from the Windows administrator to the EPiServer administrator. If you enable Impersonation, you must give all users NTFS access to the configured directories anyway. Otherwise you still let the Windows administrator control access.

Search Results Display Files that Users Don’t Have Access to

Make sure you configured the search template to use Unified File System for search instead of working directly against Index Server.

Error Message "TransmitFile.." when Downloading Secured Files

The download manager uses a function added in .NET Framework 1.1 SP1. Make sure that you upgrade your system. You can also download a hotfix instead, if don’t want to upgrade to SP1 as advised by the EPiServer Installer. This hotfix is available at http://www.episerver.com/download/hotfix/kb823409.zip.

Appendix A: web.config Used in Examples

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
 <configSections>
    <sectionGroup name="episerver">
      <section name="unifiedFileSystem" allowDefinition="MachineToApplication"
allowLocation="false"
type="EPiServer.FileSystem.Handler.ConfigurationHandler,EPiServer" />
    </sectionGroup>
  </configSections>
<episerver>
    <unifiedFileSystem>
    <mappings>
        <map fromPath="/upload/documents" toPath="/Secure"
type="EPiServer.FileSystem.Mapping.SimpleMapping,EPiServer" />
        <map fromPath="/upload" toPath="/pages"
type="EPiServer.FileSystem.Mapping.PageDirectoryMapping,EPiServer" />
    </mappings>
      <handlers>
        <handler pageDirectory="False" virtualName="upload"
virtualShare="False"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="/upload" />
        </handler>
         <handler pageDirectory="False" virtualName="images"
virtualShare="False"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="/images" />
        </handler>
         <handler pageDirectory="False" virtualName="Secure"
virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="c:\inetpub\SecureFiles" />
        </handler>
        <handler pageDirectory="False" virtualName="Extranet"
virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
          <customSettings PhysicalPath="c:\inetpub\Extranet" />
        </handler>
         <handler pageDirectory="True" virtualName="pages" virtualShare="True"
type="EPiServer.FileSystem.Handler.NativeFileSystem,EPiServer">
              <customSettings PhysicalPath="c:\inetpub\Pages" />
         </handler>
      </handlers>
    </unifiedFileSystem>
  </episerver>
(..other configuration settings)
</configuration>
```

Appendix B: Configuration Documentation

Schema

```
<episerver>
    <unifiedFileSystem>
   <mappings>
        <map fromPath="<path>" toPath="<path>" type="<mapper type>" />
   </mappings>
      <handlers>
        <handler pageDirectory="<True|False>" virtualName="<name>"
virtualShare="<True|False>" type="<Class,Assembly>">
          <customSettings <custom key>="<custom value>" />
        </handler>
      </handlers>
    </unifiedFileSystem>
  </episerver>
```

Element: *map*

Attribute	Values	Description
`fromPath`	Relative path	Used to redirect all requests from this base path
`toPath`	Relative path	Used to redirect all requests to this base path
`type`	Class,Assembly	The class that will handle incoming mapping requests

Element: *handler*

Attribute	Values	Description
`pageDirectory`	True or False	Configures a folder to be used only as the special page folder container
`virtualName`	A name of the folder	The root name of the folder in a site, must match the actual folder name when virtualShare is set to False. Otherwise it can be any name that does not match a existing folder name.
`virtualShare`	True or False	If EPiServer should activate security and if files should be delivered through EPiServer and not IIS.
`Type`	Class,Assembly	The type of filesystem that will handle this folder, for example NativeFileSystem if the filesystem is a regular folder in Windows.