Migration Guide

A Project Guide for Migrating WSM Solutions

1 Purpose

This document tells you how to design a WSM solution that you can migrate, and how to migrate the site. The information in this document is independent of the release. It focuses on general design and management principles.

2 Designing for Change

This section tells you how to set up your WSM project so that you can extend and migrate it later. The same guidelines and practices that help ensure a successful migration project also help you in your development project. We recommend that you include these in your development effort, so migration will be easier and safer later on.

2.1 Respecting the File Structure

In WSM, everything is stored in the internal repository: most of WSM itself, and your entire WSM solution. For easy migration, it is vital that you understand what is stored where, and that you respect this. If you place your project files in the wrong locations, they may be deleted in the migration process.

2.1.1 WSM System Folders

The WSM system folders are completely overwritten in the migration process. Do not place any files in them, or modify the existing files, unless you absolutely have to. If you do this, keep a list of what you did, because you must migrate these files manually.

/system

This folder contains the WSM system files. Do not modify any of these files.

/libs

This folder contains WSM's library files. If possible, do not modify any files here. In some rare instances, you have to modify these files, for example if you extend the media library. If you feel that you need to adjust these files, try overlays or an alternative to it.

2.1.2 Project-Related Folders

The project-related folders in WSM contain basic functions and settings that you can (and probably will) modify for your project. When you migrate, the update installer leaves most of these files untouched, and only adds a few settings if the new release requires them.

/apps

This folder contains WSM's project-related functions. You may modify the existing functions, or add your own functions here. This

folder also contains the components, and this is where you can store your new components. Migrating does not affect this folder.

/etc

This folder contains the functions that are displayed in the “Miscellaneous” tab in the WSM Development Environment. You can modify these functions, add new ones, or remove the existing ones. Migrating will not overwrite these changes.

/config

This folder stores your main configuration settings. Migrating may add some new entries to these files, but does not overwrite or modify existing information.

/bootstrap

This folder contains some key configuration files to start up WSM. Migrating may add some new entries to these files, but does not overwrite or modify existing information.

2.1.3 Content Folders

WSM stores the content and the users in one folder each. This allows you to separate them from the layout and the functions of your WSM solution. Migrating does not affect these folders.

/content

This folder contains the entire content of the WSM solution.

/access

This folder contains the user access information.

2.1.4 Other Folders

/tmp

This folder stores temporary files. It is not relevant for the migration process.

/classes

This folder contains the compiled java classes and jsp files. It is not relevant for migration.

2.2 Overlaying Library Files

In WSM, the libraries folder stores several useful functions for both WSM's internal works and for your project. Overlays provide a way to adapt these libraries to your needs, but they may introduce significant migration effort. This section tells you how to best use overlays, and which alternatives exist.

2.2.1 How Overlays Work

All library files are stored in the folder /libs. When WSM accesses a library file, it checks if there is a file with the same name and relative path in the /apps folder. If it is, then WSM uses this file instead of the library file.

2.2.2 "Safe" Overlaying

Because libraries can change without notice between releases, overlaying files is never entirely safe. However, in some cases, the simplicity of overlaying a file justifies the possible migration effort. You can overlay the following files without having to expect major migration problems:

1. Configuration files are usually straightforward to overlay. The main configuration file of a library is usually named "modconfig.any". There may be additional configuration files.
2. Some libraries, such as the workflow library, contain default code that you can overlay. The code files are documented, and may consist only of empty classes and functions which you can overlay with your own code.
3. Refer to the library documentation to find out which files you can (or have to) overlay.

Note: FileNet does not guarantee that any overlays work as expected after migration. However, the above items have a high probability of doing so, and are fairly easy to adapt if they do not.

2.2.3 Unsafe Overlaying

If you overlay program code or a number of files, changes to the library files may easily break your overlay, or the library, or both. Also, future versions of WSM may not be compatible with your modified library.

If you decide to overlay program code or several files, proceed as follows:

- If you overlay program code, document the code well, and say how you can test whether the overlay works. This will make migration easier.

- Only overlay as few files as possible, and never overlay a file that you do not have to modify. Your overlay files will hide any changes to the underlying library files.
- Keep a list of the reasons you had for introducing the changes. It is possible that an update to the next WSM release solves some of these reasons, so you can remove the overlay.

Note: If you overlay several library files or substantial amounts of program code, you are in effect taking over the development and maintenance effort of the library. It may be difficult or impossible to adapt your overlay files to the new library files, and you (and WSM) may not be able to use new library functions. For this reason, it may be better to start your own development effort instead of overlaying the library.

2.2.4 Alternatives to Overlaying

As an alternative to overlaying, consider putting your functions into templates or components. You may use all of the library functions there, and adapt them to your needs. However, you can still update the libraries without expecting any problems.

2.3 Quality Assurance

Testing is a central part of both development and migration projects. If you create tests during the development phase, you can re-use them in the migration phase to make sure that everything still runs as expected. There are several approaches to testing (the exact definitions may vary):

Smoke Test

A smoke test is generally a quick, rough, automated test that checks if the more obvious features of the test candidate work as expected. Smoke tests are used to quickly decide if a test candidate qualifies for thorough testing. An automated smoke test may run in under 30 minutes.

Sanity Test

A sanity test is a quick, but usually deeper test of those areas where you expect problems with a particular test candidate. Sanity tests may or may not be automated.

Regression Test

A regression test aims to test all the features of the product in depth, or at least all the features that are important to you. As such, it is relatively difficult and time-consuming to perform. If it is

automated, you can re-use it for later project phases, because the things that worked once will usually work in the same way in future releases. An automated regression test may require a few hours to run.

We recommend that you develop a regression test framework when you develop your WSM solution, and then maintain the framework so that you can use it to test migration and update projects.

Performance Test (Stress Test)

A performance test makes sure that the application you test behaves well under stress and delivers the required performance.

In WSM, always test the following areas for performance:

- Everything that involves WSM's internal search function. Improper use of the search function may lead to dramatic performance issues.
- Everything where you iterate through large parts of the content, such as a navigation bar.

Defining Success

In a complex development project, success may not be trivial to define. Even after extensive testing and bugfixing, there may still be bugs, such as:

- Very rare bugs that you cannot track down.
- Bugs in WSM.
- Bugs that are not severe enough to justify putting the entire migration project on hold.

In the process of the migration project, define the criteria that your WSM solution has to meet to switch it live. Make sure that these criteria are testable, and that you have automatic, comprehensive tests for them if possible.

2.4 Setting a Goal for Migration Projects

This guideline covers some development techniques that require manual migration effort. As part of your development project, try to limit the use of these techniques. If you need to use some of them, set a maximum constraint for future migration efforts, and try to factor the migration times into the overall equation.

For example, overlaying a library file may be a fast way to introduce an additional function, but it requires testing and maintenance for every update. So in the end, you may prefer a

solution that requires more development time, but no migration effort.

As a very rough base estimate, you may require five days for a migration project. You will spend most of this time on checking the WSM solution for migration issues, and on testing. If you are familiar with migrating a WSM solution and if testing is straightforward, you may migrate in one day (or less). However, this is mainly the case for single-purpose Web sites with few or no custom functions.

3 Deciding to Migrate

New releases of WSM introduce new features, fix known bugs, but sometimes also introduce new constraints on your WSM solution or disable outdated features. This section tells you hot to gather information about the benefits and efforts of migration.

3.1 Gathering Information

Before you address the migration project, you need to gather some information. The following list helps you to find the relevant information and ask the right questions:

- Make a list of your current problems and limitations. This may include items such as “performance is sometimes unsatisfactory”, “I have to move to a new application server”, and so on.
- Read the release notes carefully and find out which of your problems are solved by migrating, and which new issues might be introduced.
- Read the migration section in the official release documentation of the new release to make a first estimate of the migration effort.
- Read the rest of this document carefully and see if any of the proposed steps may cause problems. Make sure that your project adheres to the standards proposed here, so you can migrate it largely automatically.
- Make sure that you fulfill the hardware requirements for the new release, and that the new release supports your platform.

3.2 Estimating the Effort

Migrating a WSM project might bear some risks in the form of delays, incompatibilities and additional development effort. To estimate these risks, use the following core questions:

- Read the section “Backward Compatibility” in the release notes. Does the new release phase out technology that you rely on? Does it make changes to technology you use?
- Do you have substantial amounts of data that you have to migrate manually?
- Do you have the necessary infrastructure for a safe migration process? The safe migration allows you to make sure in advance that the migration succeeds technically, and allows you to revert to the old status quo if any problems arise once the new release is live.

- Do you have a quality assurance process in place? Quality assurance means that you find issues before your customers do, and before they cost you.
- Do you have the necessary management resources and project resources for a solid migration project?
- Does your project use overlays? Overlays may seriously increase the migration effort, especially if your overlay files contain program code that has to be adapted.
- Did you extend the WSM Web application or add other servlets? If so, you must migrate these changes to the new WSM version manually.
- Do you plan to introduce new features to your WSM solution at the same time as migrating? While the new features require additional effort, you may save time on combining the quality assurance for the features and the migration.

3.3 Deciding on the Migration Path

There are three paths to migrating. The first two require a similar effort and provide the same level of safety. The third path offers less safety and is recommended only for quick updates on small, non-vital WSM solutions that can afford some down time.

- Path 1: Copy the old WSM system to the migration server, and then update it with the update installer. This is usually faster and simpler for quick updates of WSM's core functions, and minimizes manual effort.
- Path 2: Install the new release of WSM on the migration server, and then migrate your project data to the new platform. This is more thorough, more efficient for comprehensive updates, and provides a clean new platform. However, you have additional manual effort to configure the new platform and migrate the content, settings, and some additional files.
- Path 3: Take the server offline, run the update, and switch the server live again. This is not recommended for vital or complex WSM solutions.

3.4 Timing

There are two considerations about the timing of your migration project.

Coordinating with Enhancements

If you plan to enhance your WSM solution, it may make sense to coordinate this with a migration project. This has the following advantages:

- You are developing on the latest version of WSM, which offers more features and better stability and performance.
- You only need to test once.
- If the migration process involves any inconvenience for the users, you only need to manage this process once.

Note: If you develop and migrate at the same time, testing is essential. If you feel that you do not have ample resources for testing, it may be better to do one thing after another.

Coordinating with WSM Beta Releases

If you migrate to a WSM release that is still in the Beta phase, you have the following advantages and potential problems:

- You will encounter more bugs.
- The bugs that affect your WSM solution are found and fixed during the beta phase.
- Your input finds its way into the product easier and faster.
- If you require FileNet to fix a bug outside of the Beta phase and the normal release process, this may be slower, more difficult, and more expensive.

Coordinating the Launch Date

When you set a launch date, consider the following factors:

- Make sure that the launch date does not collide with major content updates. During the launch, editing content may be impossible for some time.
- Do not place the launch date at the end of a week. If there are problems, you may need additional time, and you may need the help of other people.

4 Path 1: Updating an Existing Instance

To update an existing WSM solution, create a copy of it and update it. The WSM update installer will perform most of the steps below, and some steps are optional.

4.1 Copying the Existing WSM Solution

Copy the existing WSM solution to a new location, so that it runs in parallel with the live site. You now have two identical WSM setups: One that runs live and one that you migrate and test.

4.2 Running the Update Installer

The update installer guides you through most of the update procedure. Some optional steps are not included.

4.2.1 Updating the Servlet Engine

The servlet engine is the technical component of the Web server that runs Java applications, such as FileNet WSM. In the standard setup, WSM installs its own servlet engine, but you may also use a third-party servlet engine.

Automatically Updating the Servlet Engine

If you migrate from version 3.5.0 or version 3.5.1, the update installer automatically updates the WSM Servlet Engine, so you can proceed with the migration. This process is largely automatic, and the update installer guides you through it.

Manually Updating the Servlet Engine (optional)

For versions 3.5.2 and higher, you do not need to update the Servlet Engine. If you wish to do so, you have to do this manually, as follows:

1. Install the new Servlet Engine. You can do this in the Tools section of the WSM installer. The new Servlet Engine is now installed next to the existing one.

2. Copy the .war files from the old Servlet Engine to the new Servlet Engine. These files are located in the folder /server/webapps. Typically, the directory contains the files author.war, publish.war, and services.war.

3. In the file /server/etc/server.xml of the new Servlet Engine, register the files. Copy the <web-app> entries from the old Servlet Engine.

4. Shut down the old Servlet Engine, and start the new one. Because both engines run on the same address, you cannot use them at the same time.

After you have updated the Servlet Engine, you can proceed with the migration on the new Servlet Engine.

4.2.2 Updating the Web Application

The WSM Web application contains all the code that makes WSM run and that is not stored in the ContentBus.

The update installer automatically updates the WSM Web application to the newest release. The old release is completely overwritten by the new program.

Note: If you have specified additional or custom library files in the file manifest.mf, the update installer adds these files to the classpath. In this way, the libraries are still available for your project after the file manifest.mf is overwritten by the update.

4.2.3 Updating WSM

The update installer automatically updates WSM to the current release. To do so, the update installer overwrites the following parts of the ContentBus:

- The /system folder. This folder contains WSM's internals. There are usually no user-created files in this folder.
- The /libs folder. This folder contains WSM's library files. If you have modified any of the libraries, the update installer will overwrite it.

The update installer also adds configuration entries to the following files in the ContentBus if required:

- config/delivery/scripting.xml
- config/delivery/reqriter.xml
- config/delivery/delivery.xml
- bootstrap/config.xml

4.3 Testing

Test the WSM solution using your existing test framework if possible, or create new tests to find out if the WSM solution runs smoothly.

4.4 Going Live

When the tests are successful, switch the new WSM solution live. Be prepared to revert to the old WSM solution if problems occur that you did not locate during testing.

5 Path 2: Migrating to a New Instance

To migrate to a new instance, you set up a new instance of WSM and migrate your content over to it.

5.1 Installing WSM

Install the current release of FileNet WSM on a server, so that runs in parallel to the existing server. You may install the new release on a separate server, or on the same server as the current WSM solution.

We recommend that you use the server that will run the new WSM solution. This makes sure that the WSM solution runs perfectly well in the real setting (or as close to the real setting as possible).

To install the new release, perform a full installation using the WSM installer. Refer to the installation guide of the current release for more information.

5.2 Migrating the Content

Create packages of all the content and project-related data that you want to migrate to the new release. You need to migrate the templates first, because migrating the content requires that the templates exist on the target platform. Proceed as follows:

1. Create a package that contains all the template definitions of your Web site.
2. Create one or more packages with the content of the Web site.
3. Install the templates on the new Web site.
4. Install the content on the new Web site.

Note: This procedure is fairly straightforward, but keep in mind that you may have a lot of content on your WSM solution, and plan accordingly.

5.2.1 Areas to Migrate

To migrate a standard WSM solution, migrate the following areas:

Application(s)	An application contains the templates, components and the code for your Web site. It is a folder in the **apps** folder, such as /apps/myApplication.
Web site content	The Web site content stores all the pages of your Web site, including the page's meta information and all the content that authors have added to

	the page. It is stored in the **content** folder, under the same name as the application, for example /content/myApplication.
Media library, package definitions, and so on	WSM's etc (or "Miscellaneous") folder contains several tools for Web site administration, such as the Media Library, the package definitions, and workflows. To migrate the data stored in these tools, migrate the /apps/etc folder.
Configuration	WSM stores its configuration settings in the folders /config and /bootstrap. If you run the new WSM instance on the same infrastructure as the existing one, you can copy the files over.
Access information	WSM stores the user and group information in the folder /access. When you migrate, copy this folder to the new instance.

If you have modified files outside of these areas, you have to migrate and test them manually. The other areas may change without notice between releases, so there is no guarantee that your modifications still work on the new instance.

5.2.2 Using CRX Packages (recommended)

WSM stores its content in a CRX repository. You can use the CRX content zipper and the CRX content loader to migrate the content.

By default, the CRX repositories for the authoring and publishing instance are available at localhost:4402/crxauthor and localhost:4402/crxpublish, respectively. The repository structure mirrors WSM's internal structure, but contains additional information (such as the version history).

Refer to the CRX user guide for more information about the content zipper and content loader.

Note that WSM uses its own user management and versioning. You do not have to migrate the CRX version history or the CRX user data.

5.2.3 Using WSM Packages (not recommended)

WSM's package mechanism allows you to migrate WSM content (Web site content as well as internal files) from one WSM instance to another.

WSM packages have the following limitations:

4. You cannot migrate versioning information. After migration, authors can no longer restore previous versions of a paragraph or page.

5. Because the publishing environment does not have an administration interface, you cannot migrate the published Web site, but have to re-publish the pages.

For more information on how to use WSM packages, refer to the WSM user guide.

5.3 Migrating the History and Logs

Because packages do not store any history information, you have to migrate the files that contain it and some other files manually from the file system of the old server to the new server.

You may decide not to migrate some of these areas. See below for details.

Page Activation History

The page activation history stores page activation information, such as when the page was last updated and by whom. WSM also needs this information to find out the current status of the page. To migrate this information, copy the folder /history from the file system of the existing WSM solution to the new solution.

If you do not migrate this information, the activation state of the pages is lost. Authors cannot see whether a page is already activated.

Logs

If you need to migrate the log files, copy the folder /logs from the file system of the existing WSM solution to the new solution.

Whether you need to migrate the logs depends on the type of information you extract from them. If you use the logs only for debugging, you may prefer to start with empty logs for the new instance. If you analyze user behavior, you may need to migrate the logs.

Search Index

To migrate the search index, copy the folder /search from the file system of the existing WSM solution to the new solution. Note that WSM creates a new search index if you do not copy the existing one. This provides you with a fresh new search index, but may require extensive time if you have a large WSM solution.

5.4 Testing

Test the WSM solution using your existing test framework if possible, or create new tests to find out if the WSM solution runs smoothly.

5.5 Going Live

When the tests are successful, switch the new WSM solution live. Be prepared to revert to the old WSM solution if problems occur that you did not locate during testing.

6 Path 3: Quick Update

The quick update path is simpler and faster, but offers less safety and fewer options if the migration does not work. Use this only if:

- Your WSM solution is simple, and does not use advanced or complex scripts.
- You can afford to take it offline while you migrate.
- In case of problems, you can afford to keep the WSM solution offline until you fix the problems or revert to a backup.

Follow these steps for a quick migration:

1. Make a backup of the current live instance(s). Test the backup procedure and the backup – a working backup is your only safeguard if the migration fails.
2. Put a "server down for maintenance" note in the document root of your server. The server is now officially down. Make sure that you still have access to the WSM solution, for example by directly connecting to the Web engine.
3. Run the update installer on all WSM instances. The instances are now automatically updated to the new version, while the content and custom developments remain untouched.
4. Test the WSM solution under the new release. If the test is positive, switch the WSM solution live. If it does not work and you cannot fix it in time, revert to the backup.

7 Special Migration Issues

In addition to the normal migration procedure, some areas need special treatment. This section describes some of these areas.

Note: In this document, version numbers refer to all minor versions as well, unless indicated otherwise. That is, “WSM 4.0” indicates versions 4.0.0, 4.0.1 and so on. “WSM 4.1” indicates versions 4.1.0, 4.1.1, and so on. “WSM 4” refers to all version 4 releases (4.0.0, 4.0.1, 4.1.0, and so on).

7.1 Migrating from 3.5 to 4.0

With release 4.0, WSM by default uses CRX as the repository instead of the ContentBus. If you update an existing installation, the update installer keeps the ContentBus repository. If you migrate to a new WSM installation, WSM uses the CRX repository.

Because migrating from the ContentBus repository to a CRX repository can lead to compatibility issues, we recommend using an update installation. The extent of the compatibility issues depends on the project. Some projects may work as they are with a CRX repository, while others may require a major re-engineering effort.

7.2 Migrating from 4.0 to 4.1

WSM 4.1 stores the data in a CRX 1.2 repository. With release 1.2, CRX switches its default database to Apache Derby. Derby is an open-source Java database.

WSM uses so-called Persistence Managers to store its content in a database. The default Persistence Manager of WSM 4.1 stores the data in the Derby database.

7.2.1 If You Use the CRX Repository

In a default installation, WSM stores the content in a CRX repository. To migrate to 4.1, use migration path 2 (migrating to a new instance). Because of internal changes in WSM and CRX, you cannot use the update installer to migrate from 4.0 to 4.1.

Note: If you use a default WSM 4.1 installation, the migrated WSM instance now runs with the Derby Persistence Manager.

7.2.2 If you use the ContentBus Repository

If you have updated an existing WSM 3.5 installation using migration path 1, then your WSM 4.0 installation uses the ContentBus repository. It does not use CRX.

If you use the ContentBus repository, you can use all migration paths.

7.2.3 Compatibility

The following WSM tools are compatible with previous releases.

7.2.3.1 WSM Development Environment

The WSM Development Environment that comes with release 4.1 is fully backward compatible with all 4.x and 3.5.x releases. We recommend that you use it instead of the older versions. The development environment is not compatible with Communiqué 3.0.x and earlier.

In the Path field of the Connection window, you need to specify the instance prefix. For example, for an authoring instance, the path may look as follows:

```
/author/cqde
```

7.2.3.2 WSM Manager

The WSM manager that comes with 4.1 is fully backward compatible with earlier WSM and CRX releases, and you can use it to manage those as well.

7.3 Stopping Content Updates

While you can perform a smooth switch from the old WSM solution to the migrated site, there will be a delay before the switching, during which authors cannot update content. The shorter you need this delay to be, the more effort you have to put into your migration project.

Stopping Content Updates for Testing

The easiest way is to stop content updates as soon as you have moved the content to the new platform, and before you start testing:

- You can save some time by updating the content just before you enter the testing phase.
- The time you need for testing depends on how far you have automated your tests.
- Be prepared that migration may fail, so you may not have any updates for a continued time, or you may cancel the migration effort and try again later.

Updating Content after Testing

If you cannot afford to stop content updates while you test, devise an automatic or at least standardized way of updating the content from your current WSM solution to the new WSM solution.

When you test the new WSM solution, make sure to test the content update procedure as well.

7.4 Migrating Live Content

If you store content on the live instance, such as forum data, make sure that you migrate this data as well. You may have to transfer the data manually, or you can automate this step.

Make sure that you test this procedure thoroughly. If necessary, stop updates to the live content while you migrate the content. This will only take a short time, but users may feel irritated if content they have entered suddenly disappears.

7.5 Modified Libraries

If you have modified or extended WSM's library files, such as the media library, you have to migrate them manually. To do so, proceed as follows:

1. Create a package of the library you have modified.
2. Install the package on the new site.
3. Test whether the library works and, if necessary, if it still integrates well with the other libraries and WSM functions.

Note: Do not modify the library files if at all possible. If it is necessary, use templates, or (if you make only small adjustments) use overlays.

7.6 Overlays

An overlay is a file that replaces a file in the WSM libraries. Because the library files may change without notice between two releases, the overlay may not work with the new library.

Note: We recommend not to use overlays if possible. If necessary, restrict overlays to few and simple modifications, or to modified configuration files. If possible, use an alternative to overlaying.

In the migration process, overlay files are automatically migrated to the new platform, while the underlying library files are updated. This may cause the following behaviors:

- The new library and the overlay work together, so everything works as expected.
- The library was modified and does not work together with the overlay file. The library does not work anymore.
- The library was modified, but the overlay overrides the changes. In this case, you may not see some or all changes in the library and you may not have access to the new

library functions. WSM functions that use the new library functions will not work anymore.

Note: You can use a so-called “Diff tool” to find out which library files have changed. You may notice that a large number of library files changes for each release. If you track these changes, you may find out where you need to modify your overlay files. Even if the files you have overlaid did not change, we recommend that you test all of your overlays thoroughly. Note also that some changes may not reflect in the Diff tool, such as changes to include files.

While overlays are migrated automatically, you have to test them thoroughly and test other parts of WSM that use the libraries you have overlaid. If you encounter any issues, you have to adapt the overlay files to the new libraries.

In the case of configuration files, adapting the file is usually straightforward. If you use more complex overlay techniques, adapting the overlays can be a small migration project of its own.

7.7 Modified Web Application

If you have modified the WSM Web application, for example by adding new servlets, you have to migrate these changes manually:

1. Move your servlets to the new server.
2. Adapt the new server’s web.xml configuration file.

Note: We recommend that you do not modify the WSM Web application if possible.

7.8 Hotfixes

A hotfix is a small patch for a pressing software bug that is not part of the official release process, and as such is not subjected to the usual quality assurance process. FileNet does not give any guarantee that a hotfix performs as expected, and you use hotfixes entirely at your own risk.

Applying WSM Hotfixes

WSM hotfixes are typically distributed as package files that you can import into WSM. If they are not, they contain installation instructions.

Migrating WSM Hotfixes

When you migrate, the update installer removes any existing hotfixes. This is usually the best option, because the improvement made by the hotfix is typically included in the new release. Also, existing hotfixes may not work with new releases of the software, or they may block some of the functions of the new release.

Before you migrate to a new release, check if there are hotfixes available for this release and check if you need to install any of them. FileNet's policy is to recommend installing a hotfix only if you are certain that you need it. Again, when you install the hotfix, you do so at your own risk.

7.9 Going Live

Typically, you develop a WSM solution on an environment with a single authoring instance. If you go live, you must migrate the content from this instance to a new environment that features both an authoring and a publishing instance.

You can migrate the content and application as described in path 2 (migrating to a new instance). However, there are a few extra issues to take care of.

7.9.1 Migrating Off the Development Instance

WSM stores the content hierarchy of a page in the file componenthierarchy in the template folder. On a development instance, this file may contain old hierarchy entries that are left over from the development phase. If these files are unavailable, WSM re-creates them using the up-to-date page structure.

When you migrate off the development instance, do one of the following:

- Delete all componenthierarchy files.
- When you create a package with the application files, exclude the componenthierarchy files from the package.

7.9.2 Publishing the New Solution

To publish a WSM solution, you move it from the authoring instance to the publishing instance. Before you do so, take care of the following issues:

1. Set up the replication agent to the publishing instance.
2. Create a package with your application (leave out the Web site content).
3. If you have changed a configuration file which has a live_publish file, copy the changes to the live_publish file.
4. Click the Recreate button to create or recreate the package.
5. Publish the package using the replication agent.
6. If you have changed the configuration or added a jar file, then restart WSM.
7. Activate the Web site content.