Oracle® Universal Content Management
File Store Provider Installation and Administration Guide
10*g* Release 3 (10.1.3.3.1)

May 2007

File Store Provider Installation and Administration Guide, 10*g* Release 3 (10.1.3.3.1)
Copyright © 2007, Oracle. All rights reserved.

Contributing Authors: Bruce Silver
Contributors: Saroj Aryal

Table of Contents

Chapter
1

INTRODUCTION

OVERVIEW

With the release of version 10*g*R3, Content Server implements a file store system for data management, replacing the traditional file system for storing and organizing content. File Store Provider exposes the file store functionality in the Content Server interface, and allows additional configuration options. For example, you can configure Content Server to use binary large object (BLOB) datatypes to store content in a database, instead of using a file system. This offers several advantages:

- integrates repository management with database management for consistent backup and monitoring processes
- helps overcome limitations associated with directory structure and number of files per directory in a file system approach
- aids in distributing content more easily across systems, for better scaling of Content Server
- allows for different types of storage devices not commonly associated with a file system, for example, content addressed storage systems and write-only devices necessary in some business uses

Caution: File Store Provider should not be uninstalled or disabled once it is installed and the default file store is upgraded. If you installed File Store Provider during the standard Content Server installation but have not yet upgraded the default file store, you can uninstall the component following the procedures in the section, Uninstalling File Store Provider (page 2-5).

This section contains the following topics:

ABOUT THIS GUIDE

This section contains the following topics:

Data Management

Content Server manages content by tracking the storage of electronic files and their associated metadata. It provides the ability for users to store and access their checked in files, any associated information, and any associated renditions. This section discusses the data management methods historically used by Content Server and how they are addressed with File Store Provider.

File Management

The first half of data management is storing electronic files checked into Content Server. With Content Server, file storage has typically been done with a traditional file system, storing electronic files in a hierarchical directory structure that include vault and weblayout directories. By using the revision information specified by the content type, security group and account (if used), files and their associated renditions are placed into particular directories within the vault and weblayout directories. For example, the primary and alternate files specified at check in are stored in subdirectories in the vault directory. The specific file location is defined to be

```
<install_dir>/vault/<dDocType>/<account>/<dID>.<dExtension>
```

where *dDocType* is the content type chosen by the user on check in, *dID* is the unique system-generated identification that identifies this revision, and *dExtension* is the extension of the file checked in. In this hierarchical model, the system uses the dDocType metadata field to distribute the files within the hierarchy established in the vault directory. Similarly, any web rendition is distributed across the hierarchy within the `<install_dir>/weblayout/groups/` directory. The web rendition is the file served out of a web server, and in the historical file system storage method, could be the native file, the alternate file, or a web-viewable file generated by Inbound Refinery or some other conversion application.

This straightforward determination of file storage location is helpful to component and feature writers, helping them understand where files are located and how to manipulate them. However, it also has the effect of limiting storage management. Without careful management of the location metadata, directories can become saturated, causing the system to slow down.

Metadata Management

The second half of data management is storing metadata associated with an electronic file. With Content Server, metadata management has typically been done using a relational database, primarily involving three database tables. Metadata enables users to catalogue content and provides a means for creating file descriptors to facilitate finding it within Content Server. For users, the retrieval is done by the Content Server, and how and where the file is stored may be completely hidden. For component and feature writers, who may need to generate or manipulate files, the metadata provides a robust means of access.

File Stores

The traditional file system model historically used by Content Server limits scalability. As data management needs grow, adding extra storage devices to increase storage space is not conducive to easy file sharing via a web-based interface. Complex, nested file structures could slow performance. Suppressing the creation of a duplicate web-viewable file when the native file format could be used could be difficult. As a consequence of dealing with large systems, for example over 100 million content items, Content Server 10gR3 has shifted to using a file store. This offers the advantages of scalability, flexibility, and manageability.

Component Features

This component installs FileStoreProvider. The FileStoreProvider component lets you define data-driven rules to store and access content managed by the Content Server. File Store Provider offers the following features:

- The ability to relocate files easily.

- The ability to partition files across multiple storage devices.
- The ability to have the web-viewable file be optional.
- The ability to manage and control directory saturation.
- The ability to integrate with third-party storage devices
- An API to use, extend, and enhance different storage paradigms.

Storing Content Using File Store Provider

When File Store Provider is implemented, checked-in content and associated metadata are examined and assigned a storage rule based on criteria established by a system administrator. Criteria can include metadata, profiles, or other considerations. The storage rule determines how vault and web files are stored by Content Server and how they are accessed by a web-server.

Compatibility with Content Server

File Store Provider is compatible with Content Server version 10*g*R3. Previous versions of Content Server are not supported.

Caution: Reconfiguring a file store provider in a production environment may cause Content Server to lose track of file locations and report files as missing. File Store Provider should be installed and tested in a development environment prior to installation in a production environment.

Installation Requirements

File Store Provider requires the following:

- An instance of Content Server 10*g*R3
- File Store Provider component (FileStoreProvider.zip)

Caution: File Store Provider should not be uninstalled or disabled once it is installed and the default file store is upgraded. If you installed File Store Provider during the standard Content Server installation but have not yet upgraded the default file store, you can uninstall the component following the procedures in the section, Uninstalling File Store Provider (page 2-5).

AUDIENCE

This guide is for system administrators of Content Server with the responsibility of installing and maintaining Content Server and managing its data.

CONVENTIONS

The following conventions are used throughout this guide:

- The notation <*Install_Dir*>/ is used to refer to the location on your system where the content server instance is installed.
- Forward slashes (/) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.
- Notes, technical tips, important notices, and cautions use these conventions:

Symbols	**Description**
💡	This is a note. It is used to bring special attention to information.
⚙	This is a technical tip. It is used to identify information that can be used to make your tasks easier.
❗	This is an important notice. It is used to identify a required step or required information.
⊗	This is a caution. It is used to identify information that might cause loss of data or serious system problems.

Chapter

2

Installation

Overview

File Store Provider can be installed as an option at the time Content Server is installed. If it was not installed during Content Server installation, the component FileStoreProvider.zip can be installed separately. FileStoreProvider.zip is located on the installation media.

Caution: File Store Provider should not be uninstalled or disabled once it has been installed and the default file store has been upgraded. If you have installed File Store Provider during the standard Content Server installation but have not yet upgraded the default file store, you can uninstall the component following the procedures in the section, Uninstalling File Store Provider (page 2-5).

At a high level, installation and configuration of File Store Provider consists of the following steps:

1. Install the File Store Provider component.
2. Restart Content Server.
3. Upgrade the default file store.
4. Edit the default or create new storage rules.
5. Create one or more new partitions if desired.

This section covers the following topics:

PRE-INSTALLATION CONSIDERATIONS

When Content Server is installed and prior to upgrading to File Store Provider functionality, the default file store replicates the traditional, hierarchical file system storage method. As such, content is designated to be stored in the vault and weblayout directories created on a file system at the time Content Server is installed. The following are some considerations prior to installing and upgrading File Store Provider:

❖ Once File Store Provider is installed and the default file store is upgraded, it should not be uninstalled. Uninstalling the component can cause error, disable check in and search functionality, and can also prevent Content Server from starting.

❖ The File Store Provider component can be disabled using the Admin Server Component Manager. This can be useful during the configuration process for correcting settings to the storage rules or provider that may be causing problems with Content Server.

❖ File Store Provider should always be tested in a development environment prior to being implemented in a production environment.

❖ Prior to installation, always back up files affected by File Store Provider, such as the provider.hda file, located in the <*install_dir*>/<*instance_dir*>/data/providers/defaultfilestore directory, the intradoc.cfg file, located in the <*install_dir*>/<*instance_dir*>/bin/ directory, and the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory.

❖ If content is checked into Content Server prior to upgrading the default file store provider, Content Server may lose track of its location.

❖ Consider whether partitions are used. Additional partitions are not required, but after File Store Provider is installed and the default file store is upgraded, the vault path in the default storage rule is set to $PartitionRoot$. A value for $PartitionRoot$ does not exist until you create a partition. Any attempt to check in content before creating a partition, changing the vault path root, or creating a new, well-formed storage rule will fail. See Understanding File Store Provider Storage Principles (page 3-8), including the sections on storage rules and path construction for additional information.

❖ Consider how you define storage rules. Storage rules are assigned to content at check-in, and determine how renditions are stored and accessed.

INSTALLATION

You can install and enable the File Store Provider component using either of the following methods:

- Installing the Component using Component Manager (page 2-3)
- Installing the Component using Component Wizard (page 2-4)

Note: The NativeOsUtils component must be enabled for File Store Provider to function. The NativeOsUtils component is installed by default with Content Server 10gR3, and is considered a system component. It should not be disabled.

Caution: File Store Provider should not be uninstalled or disabled once it is installed and the default file store is upgraded. If you installed File Store Provider during the standard Content Server installation but have not yet upgraded the default file store, you can uninstall the component following the procedures in the section, Uninstalling File Store Provider (page 2-5).

Installing the Component using Component Manager

To install the File Store Provider component using Component Manager:

1. Log in to Content Server as an administrator.
2. Select **Admin Server** from the Administration menu.
3. Click the applicable Content Server instance button.
4. Click the **Component Manager** link. The Component Manager page is displayed.
5. Click **Browse**, navigate to the **FileStoreProvider.zip** file, select it, and click **Open**. The path is displayed in the Install New Component field.
6. Click **Install**. A list of component items that will be installed is displayed.
7. Click **Continue**. Component Manager asks if you want to immediately enable the File Store Provider component or return to the Component Manager. Select the option to enable the component.
8. Restart Content Server.

Installing the Component using Component Wizard

To install the File Store Provider component using Component Wizard:

1. Start the Component Wizard by selecting **Start—All Programs—Oracle File Store Provider—<*instance*>—Utilities—Component Wizard** (Windows) or by running the *ComponentWizard* script in the */bin* directory (UNIX). The Component Wizard main screen and the Component List screen are displayed.
2. On the Component List screen, click **Install**. The Install screen is displayed.
3. Click **Select**. The Zip File Path screen is displayed.
4. Navigate to the **FileStoreProvider.zip** file and select it.
5. Click **Open**. The zip file contents that will be installed are added to the Install screen list.
6. Click **OK**. Component Wizard asks if you want to enable the File Store Provider component.
7. Click **Yes**. The File Store Provider component is listed as enabled on the Component List screen.
8. Exit the Component Wizard.
9. Restart Content Server.

VERIFYING INSTALLATION

Upon successful installation of File Store Provider, four pages, three metadata fields, five resource tables, and two database tables are added to Content Server.

The following pages are added to Content Server:

- ❖ Partition Listing Page (page 3-13)
- ❖ Add/Edit Partition Page (page 3-14)
- ❖ Edit File Store Provider Page (page 3-17)
- ❖ Add/Edit Storage Rule Page (page 3-19)

The following metadata fields are added to Content Server:

- ❖ xPartitionId (page 3-3)

- xWebFlag (page 3-3)
- xStorageRule (page 3-4)

The following resource tables are added to Content Server:

- PartitionList Table (page 3-25)
- StorageRules Table (page 3-26)
- PathMetaData Table (page 3-27)
- PathConstruction Table (page 3-29)
- FileSystemFileStoreAlgorithmFilters Table (page 3-30)

The following tables are added to the database:

- FileStorage Table (page 3-30)
- FileCache Table (page 3-30)

Resource Table Location

The PathMetaData Table, PathConstruction Table, and StorageRules Table are defined in the provider.hda file of the defaultfilestore directory. The defaultfilestore directory is located in the <*install_dir*>/<*instance_dir*>/data/providers/ directory.

The PartitionList resource table is defined in the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory. Partitions are added to this table and configured using the Add/Edit Partition Page (page 3-14) within Content Server.

Caution: Resource files should not be edited directly. Proper modification of resource files should be done within the Content Server user interface or through additional component development. For more information on component development, see the "*Working With Content Server Components*" guide.

UNINSTALLING FILE STORE PROVIDER

You can uninstall the File Store Provider component using either of the following methods:

- Uninstalling File Store Provider Using Component Manager (page 2-6)
- Uninstalling File Store Provider Using Component Wizard (page 2-7)

Caution: File Store Provider should not be uninstalled or disabled once the default file store is upgraded. You can uninstall File Store Provider only if you have not yet upgraded the default file store.

Uninstalling File Store Provider Using Component Manager

To uninstall the File Store Provider component using Component Manager:

1. Log in to Content Server as an administrator.
2. Select **Admin Server** from the Administration menu. The Content Admin Server page is displayed.
3. Click the name of the Content Server instance where the component will be uninstalled. The Content Admin Server <*instance_name*> page is displayed.
4. Click **Component Manager**. The Component Manager page is displayed.
5. Select **FileStoreProvider** in the Enabled Components list.
6. Click **Disable**.
7. Click **Start/Stop Content Server**. The Content Admin Server <*instance_name*> page is displayed.
8. Restart Content Server.
9. Click **Component Manager**. The Component Manager page is displayed and the File Store Provider component is in the Disabled Components list.
10. Select the File Store Provider component in the Uninstall Component drop-down menu.
11. Click **Uninstall**. Component Manager asks if you want to uninstall the component.
12. Click **OK**. Component Manager displays a message that the File Store Provider component was uninstalled successfully.
13. Select the link to return to the Component Manager. The Component Manager page is displayed.
14. Click **Start/Stop Content Server**. The Content Admin Server <*instance_name*> page is displayed.
15. Restart Content Server to apply the changes.

Note: Uninstalling a component removes the functionality that the component provided Content Server, but the component files are not deleted from the file system.

Uninstalling File Store Provider Using Component Wizard

To uninstall the File Store Provider component using Component Wizard:

1. Start the Component Wizard by selecting **Start—All Programs—Oracle File Store Provider—<*instance*>—Utilities—Component Wizard** (Windows) or by running the *ComponentWizard* script in the */bin* directory (UNIX). The Component Wizard main screen and the Component List screen are displayed.
2. On the Component List screen, select the File Store Provider component, and click **Disable**.
3. Restart Content Server.
4. On the Component List screen, select the File Store Provider component, and click **Uninstall**. Component Wizard asks if you want to uninstall the File Store Provider component.
5. Click **Yes**. The Uninstall screen is displayed and lists the zip file contents that will be uninstalled.
6. Click **OK**. The File Store Provider component is removed from the Component List screen.
7. Exit the Component Wizard.
8. Restart Content Server to apply the changes.

Note: Uninstalling a component removes the functionality that the component provided Content Server, but the component files are not deleted from the file system.

ENABLING FILE STORE PROVIDER LOGGING

Filestore tracing is available on the System Audit Information page of Content Server. If you are having trouble after installation, try enabling tracing by adding *filestore* to the tracing selections and viewing the server output. For more information about tracing, see the *Content Server Troubleshooting* guide.

Chapter

3

CONFIGURATION

OVERVIEW

With the release of version 10*g*R3, Content Server implemented a file store for data management, replacing the traditional file system for storing and organizing content. After the File Store Provider component is installed, the system administrator can upgrade the default file store to make use of functionality exposed by the component. Once the default file store is upgraded, the web, vault and web URL path expressions can be modified.

Note: After File Store Provider is installed and the default file store is upgraded, the vault path in the default storage rule is set to $PartitionRoot$. A value for $PartitionRoot$ does not exist until you create a partition. Partitions are not required to run File Store Provider, but any attempt to check in content before creating a partition, changing the vault path root, or creating a new, well-formed storage rule will fail. See Understanding File Store Provider Storage Principles (page 3-8), including the sections on storage rules and path construction for additional information.

Caution: Resource files should not be edited directly. Proper modification of resource files should be done within the Content Server user interface or through additional component development. For more information on component development, see the "*Working With Content Server Components*" guide.

Three other resource tables are used to define and handle file paths. The defaults for the PathMetaData Table and PathConstruction Table cover most scenarios. The StorageRules Table (page 3-26) stores the values specified when a storage rule is defined. These three tables are provider-specific, and as such are defined in the provider.hda file of the defaultfilestore directory. The defaultfilestore directory is located in the

<*install_dir*>/<*instance_dir*>/data/providers/ directory. A fourth table, the FileSystemFileStoreAlgorithmFilters Table requires a component along with java code to modify.

This section contains the following topics:

USING STANDARD FILE STORE PROVIDER VARIABLES

When installing File Store Provider, several modifications are made to the Content Server database, Content Server metadata fields, and other configuration files, allowing for possible configuration options.

This section contains the following topics:

Database Options

In some situations, content stored in a database may have to be forced onto a file system. One example would be when Inbound Refinery must have access to a file for conversion. Files forced onto a file system are considered temporary cache. The following configuration values are used to control when the temporarily cached files are to be cleaned up. Note that the system cleans up files only that have an entry in the FileCache Table (page 3-30).

Variable	Description
FsCacheThreshold	Specifies the maximum cache size, in megabytes (default=100). Once the threshold is met, Content Server starts deleting files that are older than the minimum age, as specified by the FsMinimumFileCacheAge parameter.
FsMaximumFileCacheAge	The age at which files are deleted, expressed in days. The default is 365.
FsMinimumFileCacheAge	The minimum age at which cached files can be deleted. This parameter is used in conjunction with the FsCacheThreshold parameter to determine when to delete cached files.

Content Server Options

When installing File Store Provider, several Content Server metadata fields are added and additional options are available for use in configuration files.

This section contains the following topics:

Configuring Added Metadata Fields

When installing File Store Provider, three metadata fields are added to Content Server:

- ❖ **xPartitionId**—This metadata field is used in conjunction with the PartitionList table to determine the root location of the content item files. It is recommended that this field be hidden on the UI, since the partition selection algorithm provides a value.
- ❖ **xWebFlag**—This metadata field is used to determine whether a content item has a web-viewable file. Consequently, if the system has content items that have only vault files, then removing this metadata field will cause the system to expect the presence of

a web-viewable and may cause harm to the system. The metadata field may be specified by the configuration value WebFlagColumn.

❖ **xStorageRule**—This metadata field is used to track the rule that was used to determine how the file is to be stored. The metadata field may be specified by the configuration value StorageRuleField.

Note: These metadata fields are added by File Store Provider on startup, and if deleted, will be added again when Content Server restarts. If the metadata fields must be permanently deleted, set the configuration variable *FsAddExtraMetaFields=false* in the intradoc.cfg file to disable the automatic creation of the fields. The intradoc.cfg file is located in the <*install_dir*>/<*instance_dir*>/bin/ directory.

Setting the Default Storage Directory

A *StorageDir* parameter may be set equal to a root directory, used for all partitions where the PartitionRoot column value has not been specified. In this case the storage directory and the partition name will be used to create the PartitionRoot parameter. The StorageDir parameter is set in the intradoc.cfg file, located in the <*install_dir*>/<*instance_dir*>/bin directory.

Standard File Store Provider Variables

In the provider.hda located in the <*install_dir*>/<*instance_dir*>/data/providers/defaultfilestore directory, the following parameters and classes are standard for a file system store:

```
ProviderType=FileStore
ProviderClass=intradoc.filestore.BaseFileStore
IsPrimaryFileStore=true

# Configuration information specific to a file system store provider.
ProviderConfig=intradoc.filestore.filesystem.FileSystemProviderConfig
EventImplementor=intradoc.filestore.filesystem.FileSystemEventImplementor
DescriptorImplementor=intradoc.filestore.filesystem.FileSystemDescriptorImplem
entor
AccessImplementor=intradoc.filestore.filesystem.FileSystemAccessImplementor
```

WORKING WITH THE FILE STORE PROVIDER

When File Store Provider is installed and the default file store upgraded, checked-in content and associated metadata are examined and assigned a storage rule based on criteria established by a system administrator. Criteria can include metadata, profiles, or other considerations. The storage rule determines how vault and web files are stored and accessed by Content Server and how they are accessed by a web-server. Files can be stored in a database or placed on one or more file systems or storage media. Partitions can be created to help manage storage location, but are not required.

This section covers the following topics:

- ❖ Upgrading the Default File Store (page 3-5)
- ❖ Adding or Editing a Partition (page 3-6)
- ❖ Editing the File Store Provider (page 3-6)
- ❖ Adding or Editing a Storage Rule (page 3-7)
- ❖ Understanding File Store Provider Storage Principles (page 3-8)

Upgrading the Default File Store

After the File Store Provider component is installed, the system administrator can upgrade the default file store to make use of functionality exposed by the component. Once the default file store is upgraded, the web, vault and web URL file path expressions can be modified.

Caution: File Store Provider should not be uninstalled or disabled once it has been installed and the default file store has been upgraded. If you have installed File Store Provider during the standard Content Server installation but have not yet upgraded the default file store, you can uninstall the component following the procedures in the section, Uninstalling File Store Provider (page 2-5).

To upgrade the default file store, perform these steps:

1. Log in to Content Server as a system administrator.
2. Open the Administration tray and click **Providers**. The Provider Listing Page is displayed.
3. Click **Info** in the Action column next to the DefaultFileStore provider. The Provider Information Page is displayed.

4. Click **Upgrade**. The Edit File Store Provider Page is displayed.
5. Click **Update** to submit the change. The Provider Listing Page is displayed.

Note: Do not navigate away from the Edit File Store page before clicking Update to submit the change. If you do not update, the upgraded file store provider will not be in effect.

6. Restart Content Server.

Adding or Editing a Partition

You can create partitions to define additional root paths to files managed by Content Server but requiring storage in different locations or on different types of media. You create partitions using the Partition Listing Page (page 3-13). When a new partition is created, File Store Provider modifies the PartitionList resource table in the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory.

To add a partition to Content Server, perform these steps:

1. Open the Administration and click **File Store Administration**.
2. If there are no partitions defined, click **Add Partition**. Otherwise, the Add/Edit Partition Page is displayed.
3. Enter a partition name. The name must be unique.
4. Modify the partition root, duplication methods, and any other pertinent parameters. See Add/Edit Partition Page (page 3-14) for more information.
5. Ensure that **Is Active** is enabled.
6. Click **Update**. The Partition Listing Page is displayed.

Editing the File Store Provider

You may edit the default file store provider at any time. To edit the file store, perform these steps:

1. Log in to Content Server as a system administrator.
2. Open the Administration tray and click **Providers**. The Provider Listing Page is displayed.
3. Click **Info** in the Action column next to the DefaultFileStore provider. The Provider Information Page is displayed.

4. Click **Edit**. The Edit File Store Provider Page is displayed.
5. Make the necessary modifications and click **Update** to submit the changes. The Provider Listing Page is displayed.

Note: Do not navigate away from the Edit File Store page before clicking Update to submit the change.

6. Restart Content Server.

Adding or Editing a Storage Rule

You may add multiple storage rules to the file store.

Important: Storage rules cannot be deleted. Carefully consider each storage rule before you create it.

Caution: Changing a storage rule after content has been checked in to Content Server may cause Content Server to lose track of the content.

To add or edit storage rules, perform these steps:

1. Log in to Content Server as a system administrator.
2. Open the Administration tray, and click **Providers**. The Provider Listing Page is displayed.
3. Click **Info** in the Action column next to the DefaultFileStore provider. The Provider Information Page is displayed.
4. Click **Edit**. The Edit File Store Provider Page is displayed.
5. Select **Add new rule**, or select the name of the rule to edit from the Storage Rules choice list, and click **Edit rule**. The Add/Edit Storage Rule Page is displayed.
6. Make the necessary modifications to the storage rule, and click **OK**. The Edit File Store Provider Page is displayed.
7. Click **Update**. The Provider Listing Page is displayed.

Important: If the web root used in the web URL file path defined in the storage rule is something other than the default weblayout directory defined for Content Server, you must add an alias or virtual directory in your web server for the web root used in the storage rule. Otherwise, Content Server does not know where to access the file. For information on adding virtual directories to your web server, see the documentation that came with your web server.

Understanding File Store Provider Storage Principles

When a content item is checked in to Content Server, it consists of metadata, a primary file selected by the user, and potentially an alternate file. The alternate file may also be selected and checked in by the user, and is presumed to be a web-viewable file. In a file system approach to Content Server, the primary file is stored in the vault directory at the root of the <*install_dir*>/<*instance_dir*>/, and is called the native file. If an alternate file is checked in, it is also stored in the vault, but is copied to the weblayout directory or passed to a conversion application, such as Inbound Refinery. If no alternate file is checked in, then the native file is copied from the vault directory to the weblayout directory, existing in two places. If no alternate file is checked in and Inbound Refinery is installed, a rendition of the native file could be created and stored in weblayout directory.

In a file system approach to Content Server, storing content in specified directories defines a path to the content. You can access content from a web browser by using a static web URL file path, when you know the content is in a specific location, or using a dynamic Content Server service request, such as GET_FILE, when you don't. With File Store Provider installed, content may or may not be stored in a file system. Consequently, a new approach to defining paths to the content must be taken.

Depending on how you set up File Store Provider, you may or may not have a static web URL. By using a dynamic Content Server service request, you can access content when you don't know the specific location. With File Store Provider, the static web URL is defined as the *web URL file*, and the dynamic access is simply called the *web URL*. On the File Store Provider user interface, you can configure only the static web URL file path. However, you can decide to have the static web URL done as a Content Server service request, essentially making it dynamic.

This section covers the following topics:

❖ Using Storage Rules on Renditions to Determine Storage Class (page 3-8)

❖ Understanding Path Construction and URL Parsing (page 3-10)

Using Storage Rules on Renditions to Determine Storage Class

When content is checked in, all versions of the content managed by Content Server are considered renditions. These renditions include the native file, web-viewable file, and any other files that may have been rendered by Inbound Refinery or third-party conversion applications.

Renditions are grouped together into a storage class, which determines where and how a rendition is accessed. Storage classes are grouped together into a storage rule, which defines the vault, web, and web URL path expressions, via a storage class. Additionally, a storage rule determines if a rendition is not stored, as in a webless file store, or if it is stored in a different device, such as a database rather than a file system.

The following examples illustrate how storage rules can determine where and how different content items can be stored.

Example:

A storage rule is defined as *File system only* on the Add/Edit Storage Rule Page (page 3-19) and the *Is Webless File Store* is disabled. In this scenario, the system makes a copy of the primary files and places them in the weblayout directory.

This traditional file system storage example typically offers the advantage of faster access time to content when compared with database storage. This advantage diminishes if the file system hierarchy is complex or becomes saturated, or as the quantity of content items increases.

Example:

A storage rule is defined as *File system only* on the Add/Edit Storage Rule Page (page 3-19) and the *Is Webless File Store* is enabled. In this scenario, no copy is made of the primary files and so the native files are the only renditions. Requests for web-viewable files are routed to the native files stored in the vault.

This traditional file system storage example, like the previous one, offers the advantage of faster access time to content. It also saves on storage space by not copying a version of the content from the vault directory to the weblayout directory. Instead, it redirects web-viewable access to the content in the vault directory. This is useful if most of the native files checked in are in a web-viewable format, or if Content Server is being used to manage content that isn't required to be viewed in a browser.

Example:

A storage rule is defined as *JDBC Storage* on the Add/Edit Storage Rule Page (page 3-19) and no selection is made from the *Renditions* choice list. In this scenario, both the vault and web files are stored in the database.

This database storage example offers the advantage of integrating repository management with database management for consistent backup and monitoring processes, and helps overcome limitations associated with directory structure and number of files per directory in a file system approach.

Important: When necessary, content items stored in a database can be forced onto the file system, for example, during indexing or conversion. The files on the file system are treated as temporary cache and deleted following the parameters specified in the config.cfg file located in the <*install_dir*>/<*instance_dir*>/config directory. For more information on the parameters used, see FileCache Table (page 3-30).

Example:

A storage rule is defined as *JDBC Storage* on the Add/Edit Storage Rule Page (page 3-19) and *Web Files* is selected from the *Renditions* choice list. In this scenario, the vault files are stored in the database and the web files are permanently stored on the file system.

This mixed approach of storing native files in a database but web-viewable files on a file system offers the advantages of database storage in the previous example (integrated backup and monitoring, overcoming file system limitations) for the native files, while providing speedy web access to web-viewable renditions. Like the first example, this advantage can be diminished if the file system structure is overly complex, or the quantity of files is extreme.

Understanding Path Construction and URL Parsing

The path to content stored in Content Server is defined in the PathExpression column of the PathConstruction Table. Paths are made up of pieces, with each piece separated by a slash (/). Each piece can be made of a static string or a sequence of dynamic parts. A dynamic part is encapsulated by $. A part may be calculated via an algorithm, IdocScript variable, environment variable or a metadata lookup, and can have the following interpretations:

- It may be a field defined in the PathMetaData table. If it is defined in the PathMetaData table, it may be mapped to an algorithm, for example, $dDocType$.
- If it has the prefix #env., it is an environment variable, for example, $#env.VaultDir$.
- It may be an IdocScript variable, for example, $HttpWebRoot$.

For example, the standard vault location is defined as

```
$PartitionRoot$/vault/$dDocType$/$dDocAccount$/$dID$$ExtensionSeparator$$dExtensio
n$
```

When parsed, the path expression turns into five pieces, interpreted according to the rules specified in the PathMetaData table as follows:

- **$PartitionRoot$**—mapped to the partitionSelection algorithm and uses the xPartitionId as a lookup into the PartitionList table to determine the partition root.
- **/vault/**—a string, so no calculation or substitution
- **$dDocType$**—by the PathMetaData table this is a look up in the file parameters
- **$dDocAccount$**—this is mapped to a documentAccount algorithm which takes dDocAccount and parses it into the standard Content Server account presentation with all the appropriate delimiters
- **$dID$$ExtensionSeparator$$dExtension$**—this piece has three parts:
  - *dID*—similar to dDocType, this is defined in the file parameters and is a required field
  - *$ExtensionSeparator$*—determined by an algorithm and by default it returns '.'
  - *$dExtension$*—similar to dDocType

In the standard configuration, the URL contains security and dDocType information as well as the dDocName and extension. The URL and the web location is constructed as follows:

```
…/groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/
$dDocName$.$dWebExtension$
```

The *groups* separator indicates to Content Server that the directories that follow are the name of the security group and account to which the content item belongs. Accounts are optional and consequently computed by an algorithm. After the security information, we have the *documents* separator, which is immediately followed by the dDocType. The last part of the URL is the dDocName and its format extension.

Because the URL is expected in this format, Content Server can successfully extract metadata from it. More importantly, it can determine the security information for the content item and derive the access privileges for a particular user.

The parsing guidelines have been expanded to allow for dispersion in the web directory. The *groups* separator is kept, but the *documents* separator may be replaced with *sg*. When the parse encounters the *sg* separator, it no longer assumes that the remaining part of the URL is `/sg/$dDocName$.$dWebExtension$`. Instead, the parser looks for the dispersion end marker *d*. Once the *d* is encountered, the system assumes that the following information contains the dDocName and dWebExtension as before. This means that the system can now successfully parse URLs of the form

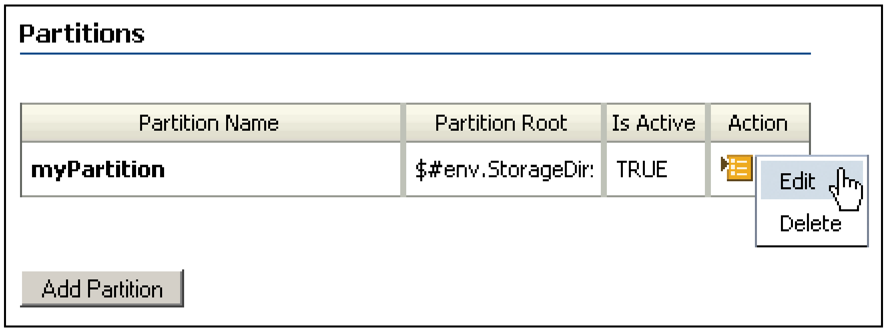
```
../groups/$dSecurityGroup$/$dDocAccount$/sg/<dispersion>/<dispersion>…/d/
$dDocName$.$dWebExtension$
```

CONTENT SERVER PAGES

When installed and enabled, File Store Provider adds the following pages to Content Server:

- Partition Listing Page (page 3-13)
- Add/Edit Partition Page (page 3-14)
- Edit File Store Provider Page (page 3-17)
- Add/Edit Storage Rule Page (page 3-19)

Partition Listing Page



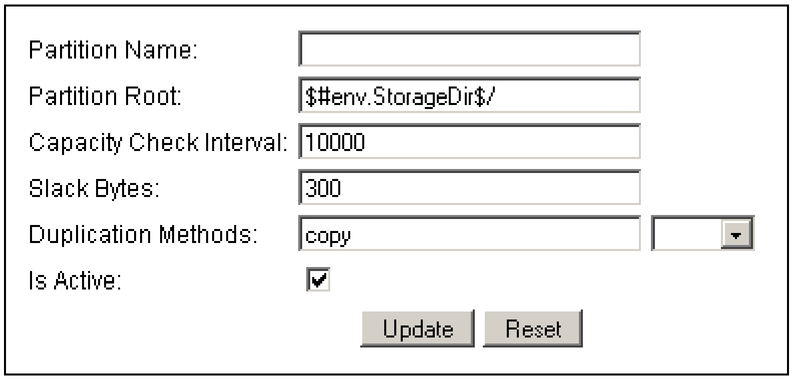
Partition Name	Partition Root	Is Active	Action
myPartition	$#env.StorageDir:	TRUE	

The Partition Listing page displays a list of all current partitions, indicating their root and status. The Partition Listing page is accessed by opening the Administration tray in the tray area of content and clicking File Store Provider Administration. Elements of each partition listed can be modified using the Add/Edit Partition Page (page 3-14), and their values are stored in the PartitionList resource table in the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory.

Element	Description
Partition Name	Displays the name of the partition as defined when the partition was created using the Add/Edit Partition Page (page 3-14). The partition name is part of the path expression used by Content Server when storing content.

Element	Description
Partition Root	Displays the root level to where content is being stored for this partition and is one of the arguments passed to the algorithm used by Content Server to choose a storage location for content. This value can be a static string, such as C:/vault, an expression, such as $#env.VauldDir$, or an Idoc Script variable, such as $HttpWebRoot$.
Is Active	Displays whether a partition is active (TRUE) or not (FALSE). Active partitions are available to store content.
Action	Displays the item action menu for each partition, from which you can choose to edit or delete the partition.
Add Partition	Clicking Add Partition displays the Add/Edit Partition Page (page 3-14), which can be used to add and activate a new partition for use by Content Server.

Add/Edit Partition Page



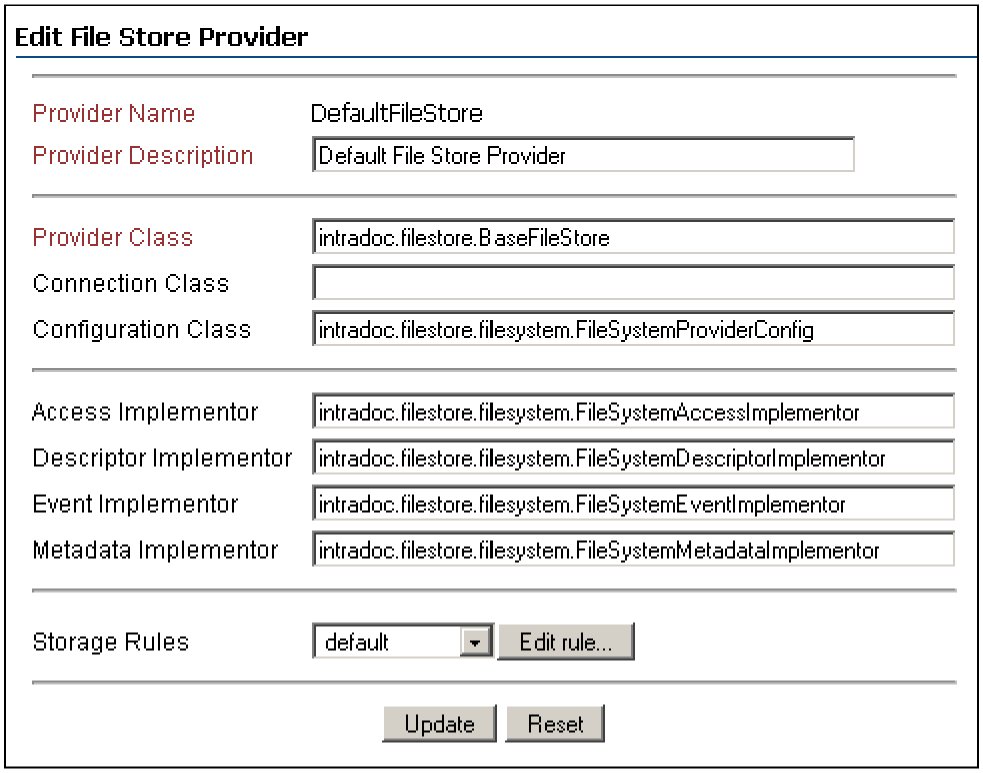
Partition Name:	
Partition Root:	$#env.StorageDir$/
Capacity Check Interval:	10000
Slack Bytes:	300
Duplication Methods:	copy
Is Active:	✓

The Add/Edit Partition Page is used to create and modify partitions used by Content Server to store content. It is accessed by clicking Add Partition on the Partition Listing Page (page 3-13). Values entered here are stored in the PartitionList Table in the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory.

Element	**Description**
Partition Name	Defines the unique name of the partition. The partition name is displayed on the Partition Listing Page (page 3-13) and is part of the path expression used by Content Server to store content. As such, it must be unique for each partition created, and has the same character limitations as Idoc Script and HTML path expressions.
Partition Root	Defines the root level of the path to where content is stored for this partition and is one of the arguments passed to the algorithm used by Content Server to choose a storage location for content.
Capacity Check Interval	Specifies the interval used in determining the disk space available for use by this partition. Expressed in seconds. This argument may not work on all platforms.
Slack Bytes	Specifies the point at which a partition is full and can no longer accept content. If the available space on the partition is lower than the specified number of slack bytes, the partition no longer accepts new content.

Element	**Description**
Duplication Methods	Specifies how native files are treated when not converted to a web-viewable rendition. For example, many image files do not require a rendition to be web-viewable. Linking to the native file instead of copying them to the web path helps manage storage space. • **copy** (default)—copies the native file to the web path. • **link**—Resolves the web path to the native file in the vault
Is Active	Specifies whether the partition is active and available for new content.
Update	Submits the information specified creates or updates the partition.
Reset	Resets the information to the previous state prior to updating the partition.

Edit File Store Provider Page



Edit File Store Provider

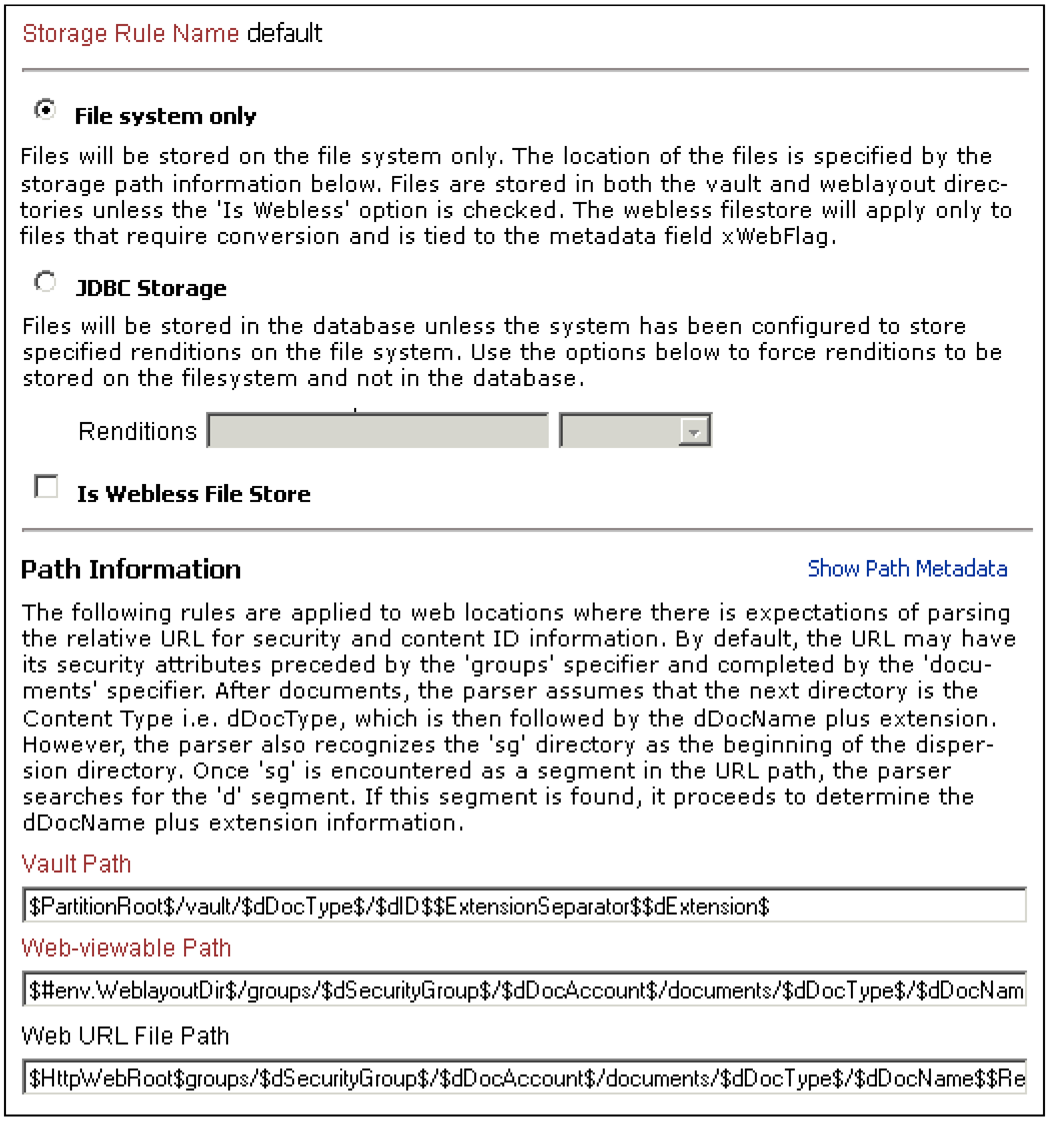
Provider Name	DefaultFileStore
Provider Description	Default File Store Provider
Provider Class	intradoc.filestore.BaseFileStore
Connection Class	
Configuration Class	intradoc.filestore.filesystem.FileSystemProviderConfig
Access Implementor	intradoc.filestore.filesystem.FileSystemAccessImplementor
Descriptor Implementor	intradoc.filestore.filesystem.FileSystemDescriptorImplementor
Event Implementor	intradoc.filestore.filesystem.FileSystemEventImplementor
Metadata Implementor	intradoc.filestore.filesystem.FileSystemMetadataImplementor
Storage Rules	default

The Edit File Store Provider page is used to modify the existing provider. It is accessed by clicking Edit on the Provider Information Page (page 3-24). The information entered here is stored in the provider.hda file located in the <*install_dir*>/<*instance_dir*>/data/providers/defaultfilestore directory. The default values will handle most storage scenarios. For more information about Providers, see the *Managing System Settings and Processes* guide that comes with the Content Server documentation.

Element	**Description**
Provider Name	Defines the name of the provider.
Provider Description	A descriptive phrase displayed on the Provider Listing Page (page 3-23) identifying the provider.

Element	**Description**
Provider Class	The path to the Java class file governing provider functionality. The default class file is *BaseFileStore*.
Connection Class	This is a path to a Java class file that is not applicable to File Store Provider. Do not enter a value.
Configuration Class	The path to the Java class file used to configure file store provider functionality.
Access Implementor	The path to the Java class file called to access content.
Descriptor Implementor	The path to the Java class file called when describing content.
Event Implementor	The path to the Java class file called when implementing an event, such as indexing or searching.
Metadata Implementor	The path to the Java class file called when needing information about content.
Storage Rules	Lists the storage rules used for the provider. Select the rule to edit, or select *Add rule* to create additional rules.
Edit Rule	Accesses the Add/Edit Storage Rule Page (page 3-19) for adding or modifying storage rules.

Add/Edit Storage Rule Page



Storage Rule Name default

(●) **File system only**

Files will be stored on the file system only. The location of the files is specified by the storage path information below. Files are stored in both the vault and weblayout directories unless the 'Is Webless' option is checked. The webless filestore will apply only to files that require conversion and is tied to the metadata field xWebFlag.

( ) **JDBC Storage**

Files will be stored in the database unless the system has been configured to store specified renditions on the file system. Use the options below to force renditions to be stored on the filesystem and not in the database.

Renditions

☐ **Is Webless File Store**

Path Information Show Path Metadata

The following rules are applied to web locations where there is expectations of parsing the relative URL for security and content ID information. By default, the URL may have its security attributes preceded by the 'groups' specifier and completed by the 'documents' specifier. After documents, the parser assumes that the next directory is the Content Type i.e. dDocType, which is then followed by the dDocName plus extension. However, the parser also recognizes the 'sg' directory as the beginning of the dispersion directory. Once 'sg' is encountered as a segment in the URL path, the parser searches for the 'd' segment. If this segment is found, it proceeds to determine the dDocName plus extension information.

Vault Path

$PartitionRoot$/vault/$dDocType$/$dID$$ExtensionSeparator$$dExtension$

Web-viewable Path

$#env.WeblayoutDir$/groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/$dDocNam

Web URL File Path

$HttpWebRoot$groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/$dDocName$$Re

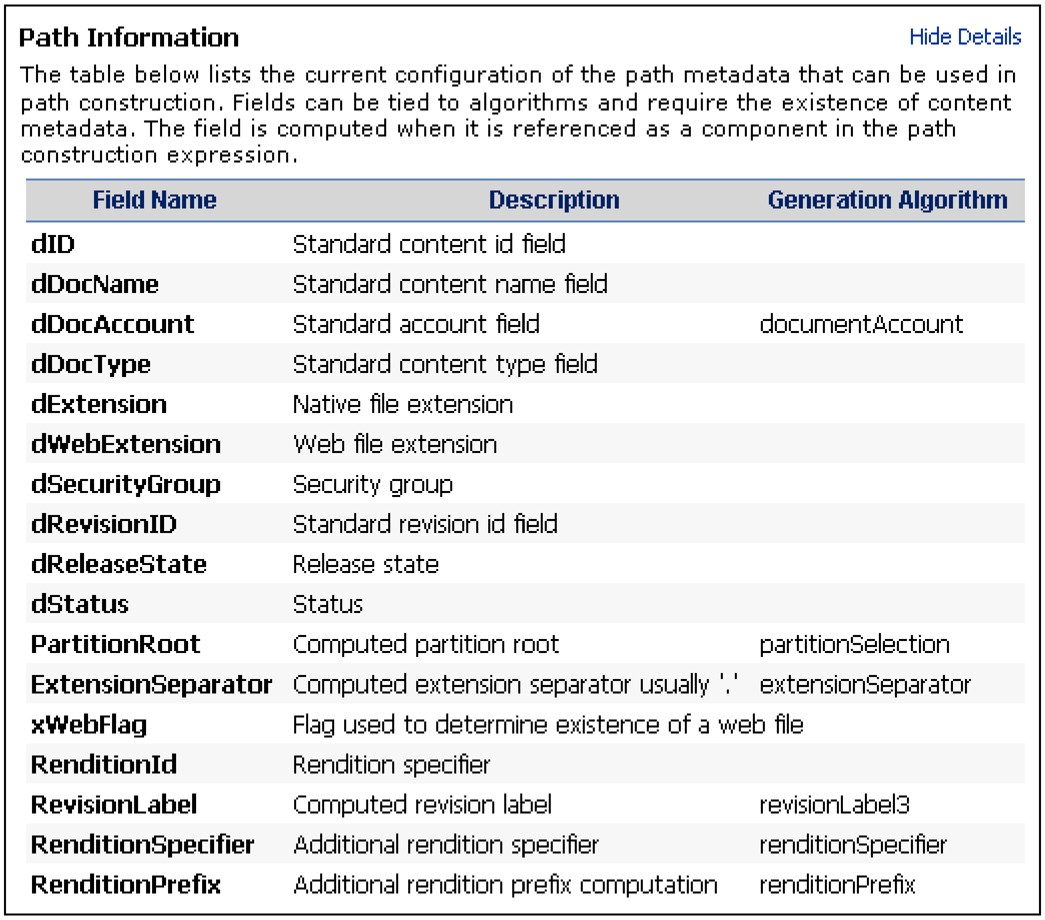
The Add/Edit Storage Rule page is used to confide how and where each provider stores content checked into Content Server. This page defines if content is stored on a file system or within a database, if a web rendition is created, and how the paths to the content are constructed. It is accessed by clicking Edit next to the Storage Rules choice list on the Edit File Store Provider Page (page 3-17).

Element	**Description**
File system only	Specifies that content checked into Content Server be stored only on a specified file system, and not in a database. This includes both the native and web-viewable files unless the Is Webless File Store option is enabled.
Is Webless File Store	Specifies that a web-viewable rendition content not be created.
JDBC Storage	Specifies that content checked into Content Server be stored only in a database, and not on a file system. This includes both the native and web-viewable files unless an option is selected from the Renditions choice list.
Renditions	Specifies a rendition to store on a file system when JDBC Storage is enabled. • No selection (default)—Both the native and web-viewables are stored in the database. • **Web Files**—Stores web-viewables on a file system and native files in the database. • **Vault Files**—Stores native files on a file system and web-viewables files in the database.
Show Path Metadata	Displays detailed information about the metadata used for constructing paths used by the provider.
Vault Path	The expression defining the path to the vault location where native content checked into Content Server server is stored for the provider using this rule.

Element	Description
Web-viewable Path	The expression defining the path on the file system to the web-viewable rendition.
Web URL File Path	The URL used to access the web-viewable rendition in a browser.

Important: If the web root used in the web URL file path defined in the storage rule is something other than the default weblayout directory defined for Content Server, you must add an alias or virtual directory in your web server for the web root used in the storage rule. Otherwise, Content Server does not know where to access the file. For information on adding virtual directories to your web server, see the documentation that came with your web server.

Path Information Detail



Path Information

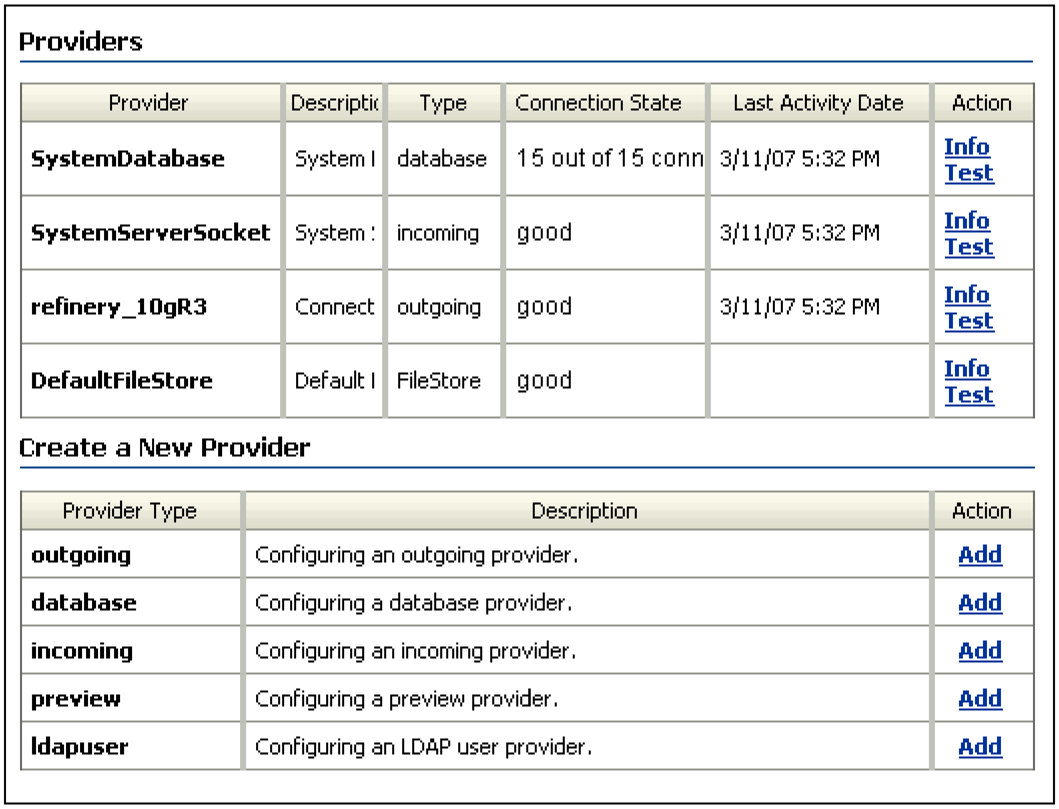
Hide Details

The table below lists the current configuration of the path metadata that can be used in path construction. Fields can be tied to algorithms and require the existence of content metadata. The field is computed when it is referenced as a component in the path construction expression.

Field Name	Description	Generation Algorithm
dID	Standard content id field	
dDocName	Standard content name field	
dDocAccount	Standard account field	documentAccount
dDocType	Standard content type field	
dExtension	Native file extension	
dWebExtension	Web file extension	
dSecurityGroup	Security group	
dRevisionID	Standard revision id field	
dReleaseState	Release state	
dStatus	Status	
PartitionRoot	Computed partition root	partitionSelection
ExtensionSeparator	Computed extension separator usually '.'	extensionSeparator
xWebFlag	Flag used to determine existence of a web file	
RenditionId	Rendition specifier	
RevisionLabel	Computed revision label	revisionLabel3
RenditionSpecifier	Additional rendition specifier	renditionSpecifier
RenditionPrefix	Additional rendition prefix computation	renditionPrefix

Detailed information about the metadata used for constructing paths used by the provider can be viewed by clicking Show Path Metadata on the Add/Edit Storage Rule Page (page 3-19).

Provider Listing Page



Providers

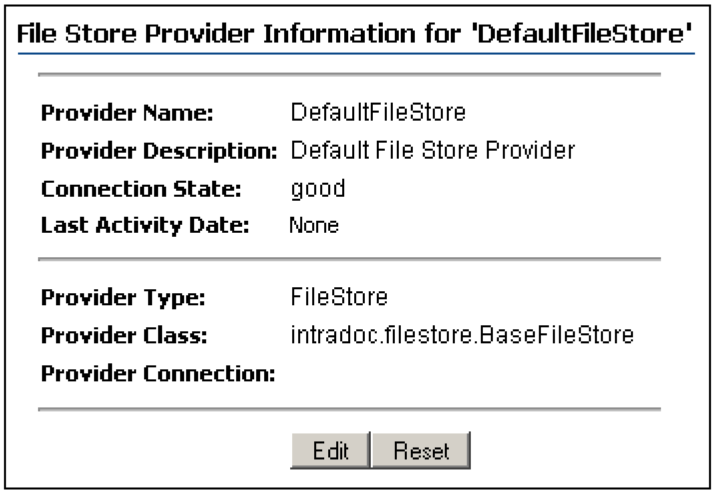
Provider	Descriptic	Type	Connection State	Last Activity Date	Action
SystemDatabase	System I	database	15 out of 15 conn	3/11/07 5:32 PM	Info Test
SystemServerSocket	System :	incoming	good	3/11/07 5:32 PM	Info Test
refinery_10gR3	Connect	outgoing	good	3/11/07 5:32 PM	Info Test
DefaultFileStore	Default I	FileStore	good		Info Test

Create a New Provider

Provider Type	Description	Action
outgoing	Configuring an outgoing provider.	Add
database	Configuring a database provider.	Add
incoming	Configuring an incoming provider.	Add
preview	Configuring a preview provider.	Add
ldapuser	Configuring an LDAP user provider.	Add

When File Store Provider is installed, the Provider Listing page of Content Server is modified to enable you to create and modify file store providers. Clicking Info next to an existing file store provider displays the Provider Information Page (page 3-24). New file store providers are created by clicking Add in the Action column of the FileStore provider type, which displays the Edit File Store Provider Page (page 3-17).

Provider Information Page



File Store Provider Information for 'DefaultFileStore'

Provider Name:	DefaultFileStore
Provider Description:	Default File Store Provider
Connection State:	good
Last Activity Date:	None
Provider Type:	FileStore
Provider Class:	intradoc.filestore.BaseFileStore
Provider Connection:	

The Provider Information page of a file store provider displays information about the selected provider, including the connection state, last activity date, and the provider type, class, and connection. Clicking Edit on the Provider Information page displays the Edit File Store Provider Page (page 3-17), where details of the provider can be modified.

FILE STORE PROVIDER RESOURCE TABLES

This section covers the following topics:

- PartitionList Table (page 3-25)
- StorageRules Table (page 3-26)
- PathMetaData Table (page 3-27)
- PathConstruction Table (page 3-29)
- FileSystemFileStoreAlgorithmFilters Table (page 3-30)
- FileStorage Table (page 3-30)
- FileCache Table (page 3-30)

PartitionList Table

The PartitionList table defines the partitions that are available for the partitionSelection algorithm. The table is defined in the fsconfig.hda file, located in the <*install_dir*>/<*instance_dir*>/data/filestore/config/ directory, and modified using the Add/Edit Partition Page in the Content Server user interface. The columns of the table are used as follows:

Column	**Description**
PartitionName	Specifies the name of the partition. This name is referenced in the path expression.
PartitionRoot	An argument passed into the partitionSelection algorithm.
IsActive	Determines if the partition is currently active and accepts new files.
CapacityCheckInterval	Specifies the interval in seconds used in determining the available disk space. This may not work on all platforms.
SlackBytes	Determines if there is sufficient space on a partition to store content. If the available space is lower than the slack bytes, the partition is deactivated and no longer used for contribution.
DuplicationMethods	Specifies how native files are treated when not converted to a web-viewable rendition. • **copy** (default)—copies the native file to the web path. • **link**—Resolves the web path to the native file in the vault Copy and Link rely on functionality of the operating system on which Content Server is installed. As such, not all methods are available on all platforms

StorageRules Table

The StorageRules table defines the rules used for storing content items. The rule specifies which path expression to use for which storage class, how content items are to be stored.

The table is defined in the provider.hda file, located in the <*install_dir*>/<*instance_dir*>/data/providers/defaultfilestore/ directory, and modified using the Add/Edit Storage Rule Page in the Content Server user interface. The columns of the table are used as follows:

Column	Description
StorageRule	The name of the storage rule. Computed from a dynamic include and stored in the *xStorageRule* metadata field of a content item.
StorageType	Determines the storage implementation. • **FileStorage—**files are stored on the file system • **JdbcStorage—**files are stored in the database
IsWeblessStore	Used to specify if system allows webless files. • **true**—by default, newly created content items do not have a web-viewable file. In certain circumstances it is necessary to insist on a web-viewable file. In such situations, an argument in the calling code can be used to specify that a web-viewable file needs to be created. Information regarding whether or not there is a web-viewable file is stored in the *xWebFlag* metadata field. • **false**—by default, newly created content items do have a web-viewable file.

Column	Description
RenditionsOnFileSystem	Used by JdbcStorage to determine if any files are to be stored on the file system instead of the database.

PathMetaData Table

The PathMetaData table defines what metadata is used to determine the location of a file. The metadata may come directly from a content item's metadata, or be calculated via an algorithm. The PathMetaData table is defined in the provider.hda file of the defaultfilestore directory. The defaultfilestore directory is located in the <*install_dir*>/<*instance_dir*>/data/providers/ directory.

The columns of the table are used as follows:

Column	Description
FieldName	Name of the field as it appears in the path expression.
GenerationAlgorithm	Specifies the algorithm used to resolve or compute the value for the field.

Column	Description
RequiredForStorage	Defines for which storage class the metadata is required. • **#all**—Both vault and web-viewable renditions require the metadata • **web**—Just the web-viewable rendition requires the metadata • **vault**—Just the native file rendition requires the metadata The field is optional for all renditions not specified. Consequently, if this column is empty, then the metadata field is optional for all renditions or storage classes. If an algorithm has been specified, this value is empty. The algorithm uses the value specified in the ArgumentFields column to dictate which fields are required.
Arguments	Optional arguments passed into the algorithm specified in the GenerationAlgorithm field.
ArgumentFields	A comma-separated list of fields required by the arguments defined in the Arguments column, and consequently required by the algorithm specified in the GenerationAlgorithm field.

PathConstruction Table

The PathConstruction table maps a file to a path. The PathConstruction table is defined in the provider.hda file of the defaultfilestore directory. The defaultfilestore directory is located in the <*install_dir*>/<*instance_dir*>/data/providers/ directory. For more information, see also Understanding Path Construction and URL Parsing (page 3-10).

Caution: The defaults provided in the PathConstruction table should work for most scenarios. This resource file should not be edited directly. Proper modification should be done through additional component development. For more information on component development, see the "*Working With Content Server Components*" guide.

The columns of the PathConstruction table are defined as follows:

Column	Description
FileStore	Specifies the storage path that is being calculated. • **web**—Path to the web-viewable • **vault**—Path to the native file • **weburl**—Generated by Content Server. Tends to be GET_FILE. • **weburl.file**—Nicely constructed URL used to access the web-viewable rendition in a browser.
PathExpression	Defines the path.
AutoCreateLimit	Specifies the depth of the directories that may be created.
StorageRule	Specifies to which storage rule this path construction belongs.

FileSystemFileStoreAlgorithmFilters Table

The FileSystemFileStoreAlgorithmFilters table is used to map an algorithm name to an implementation of the FilterImplementor interface. The algorithm can be referenced in the PathMetaData Table and is used to calculate the desired path field. The class implementing the algorithm must return the required metadata fields it uses for calculation, when the file parameters object is null. Via the ExecutionContext, the doFilter method is passed in information about the field, content item, and file store provider that initiated the call. In particular, for the file system provider, the algorithm will be passed the following information via the ExecutionContext. Bear in mind that other file store providers may choose to pass in more or possibly different information.

```
Properties fieldProperties = (Properties)
    context.getCachedObject("FieldProperties");
Parameters data = (Parameters)
    context.getCachedObject("FileParameters");
Map localData = (Map) context.getCachedObject("LocalProperties");
String algorithm = (String) context.getCachedObject("AlgorithmName");
```

The FileSystemFileStoreAlgorithmFilters table is part of File Store Provider and requires a component along with java code to modify.

Caution: The defaults provided in the FileSystemFileStoreAlgorithmFilters table should work for most scenarios. This resource file should not be edited directly. Proper modification should be done with Java code and through additional component development. For more information on component development, see the "*Working With Content Server Components*" guide.

FileStorage Table

The FileStorage table is added to the Content Server when File Store Provider is installed. It is used exclusively by the JdbcStorage storage type, when content is stored in a database. The FileStorage table contains the renditions of content items and uses the dID of the content item and rendition to uniquely identify what renditions belong to which content item.

FileCache Table

The FileCache table is added to the Content Server when File Store Provider is installed. It is used exclusively by the JdbcStorage storage type to remember which renditions have been placed on a file system. Renditions stored in a database are placed on a file system when required for a specific event, for example indexing or conversion. These files are often temporary and deleted after a specified interval as part of a scheduled event.

Chapter
4

SAMPLE IMPLEMENTATIONS

OVERVIEW

In this section, we explicitly list the contents of the tables contained in the provider definition file (provider.hda) for each of the examples. The provider.hda file does not need to be edited manually. Proper modification of the provider.hda file should be done within the Content Server user interface using the Add/Edit Partition Page (page 3-14), or through additional component development. The provided default options for other resource tables, such as PathMetaData Table, PathConstruction Table, and FileSystemFileStoreAlgorithmFilters Table, should have sufficient flexibility for most scenarios.

This section covers the following topics:

Example PathMetaData Table Options

In most of the examples, the following PathMetaData Table configuration definitions are used. The table has been trimmed of some it columns not pertinent to the examples for clarity.

```
@ResultSet PathMetaData
6
FieldName
GenerationAlgorithm
RequiredForStorage
    <trimmed columns>
dID

#all
dDocName

#all
dDocAccount
documentAccount

dDocType

#all
dExtension

#all
dWebExtension

weburl
dSecurityGroup

#all
dRevisionID

#all
dReleaseState

#all
dStatus

web
xPartitionId
partitionSelection
```

```
ExtensionSeparator
extensionSeparator

xWebFlag

RenditionId

#all
RevisionLabel
revisionLabel

RenditionSpecifier
renditionSpecifier

@end
```

Configuration for Standard File Paths

File Store Provider can be configured to place content on a file system in the standard Content Server locations.

Defining the Storage Rule

The first step is to define the storage rule. In this case, the storage rule will be of type *FileStorage*, because all content is to be stored on the file system.

Example:

```
@ResultSet StorageRules
4
StorageRule
StorageType
IsWeblessStore
RenditionsOnFileSystem
default
FileStorage

@end@
```

Defining the Path Construction

The second step is to define the path construction for each of the storage classes for the rule. In general, the last part of the path should be standard for all usage examples. If not, then Content Server will not work well with hcs* files. However, the root path can be changed without affecting functionality, assuming that changing the web URL file path root is properly acknowledged by the web server as a content server web root.

In this configuration, the vault, web and web URL storage classes need to be defined in the PathConstruction Table (page 3-29). The path expression for the vault has already been discussed in Chapter 3 (*Understanding Path Construction and URL Parsing)*. So we will only look at the web path expression, which differs from the web URL only in its root. In other words, the web path is an absolute path on the file system, while the web URL is a URL served up by a web server.

Example:

```
@ResultSet PathConstruction
4
FileStore
PathExpression
AutoCreateLimit
IsWritable
StorageRule
vault
$#env.VaultDir$$dDocType$/$dDocAccount$/$dID$$ExtensionSeparator$$dExtension$
6
true
default
weburl
$HttpWebRoot$groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/
    $dDocName$$RenditionSpecifier$$RevisionLabel$$ExtensionSeparator$
    $dWebExtension$
3
false
default
web
$#env.WeblayoutDir$groups/$dSecurityGroup$/$dDocAccount$/documents/
$dDocType$/$dDocName$$RenditionSpecifier$$RevisionLabel$$ExtensionSeparator$$d
WebExtension$
3
true
default
@end
```

The web path construction is defined to be:

```
$#env.WeblayoutDir$groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/
$dDocName$$RenditionSpecifier$$RevisionLabel$$ExtensionSeparator$$dWebExtension$
```

This is parsed into its parts and are as follows:

Path Segment	Description
$#env.WeblayoutDir$	Look up in the shared environment for the value 'WeblayoutDir'. This is defined by Content Server to be the physical root path of the weblayout directory.
$HttpWebRoot$	Alternate IdocScript variable for web URL
groups/	String
$dSecurityGroup$	Used by the PathMetaData table. This is a required field and must consequently be provided by the caller or descriptor creator. It is part of a content item's metadata information.
$dDocAccount$	This is mapped to a documentAccount algorithm which takes dDocAccount and parses it into the standard Content Server account presentation with all the appropriate delimiters.
/documents/	String
$dDocType$	Used by the PathMetaData table. This is a required field and must consequently be provided by the caller or descriptor creator. It is part of a content item's metadata information.
$dDocName$	Used by the PathMetaData table. This is a required field and must consequently be provided by the caller or descriptor creator. It is part of a content item's metadata information.

Path Segment	Description
$RenditionSpecifier$	This is provided by the renditionSpecifier, which is only of interest if the system is creating additional renditions such as thumbnails. Otherwise, this returns an empty string.
$RevisionLabel$	The revision label is provided by the revisionLabel algorithm which, depending on the status of the content item, adds a '~dRevLabel' to the path.
$ExtensionSeparator$	The extensionSeparator algorithm is used here and by default it returns '.'.
$dWebExtension$	The dWebExtension is a required field for the web and web URL storage classes and is passed in via the file parameters.

Configuration for a Webless or Optional Web Store

In this example, the previous example storage rule is configured to have IsWeblessStore set to true and consequently the web-viewable file will not be created by default. However, if the document is processed through Inbound Refinery or WebForms or any other component that requires a web-viewable, the web file will be created. The location of the files is as above in the 'standard' configuration. However, since a file may not have a web rendition, the web URL path needs to be adjusted. Also, note the use of web URL.file. This is used to compute the URL when the web-viewable actually exists. The metadata field xWebFlag is used to determine how the file is to be served up in the browser.

Defining the Storage Rule

Example:

```
@ResultSet StorageRules
4
StorageRule
StorageType
IsWeblessStore
RenditionsOnFileSystem
default
FileStorage
true
@end@
```

Defining the Path Construction

Example:

```
@ResultSet PathConstruction
4
FileStore
PathExpression
AutoCreateLimit
IsWritable
vault
$#env.VaultDir$$dDocType$/$dDocAccount$/$dID$$ExtensionSeparator$$dExtension$
6
true
```

```
default
weburl
$HttpCgiPath$?IdcService=GET_FILE&dID=$dID$
    &dDocName=$dDocName$&allowInterrupt=1&noSaveAs=1&fileName=$dOriginalName$
3
false
default
weburl.file
$HttpWebRoot$groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/
    $dDocName$$RenditionSpecifier$$RevisionLabel$$ExtensionSeparator$
    $dWebExtension$
3
false
default
web
$#env.WeblayoutDir$groups/$dSecurityGroup$/$dDocAccount$/documents/
    $dDocType$/$dDocName$$RenditionSpecifier$$RevisionLabel$
    $ExtensionSeparator$$dWebExtension$
3
true
default
@end
```

Configuration for Database Storage

To store files in the database, we need a storage rule that is of type *JdbcStorage*. By default, all content items belonging to this rule have their files stored in the database. However, even though the files are stored in the database, there is the presumption of an underlying file system and the system may need to temporarily cache a file on the file system. In particular, this may happen for indexing or for some conversions.

Tech Tip: A rule can be configured to always store renditions belonging to a given storage class on the file system. This is most useful for systems that store vault files in the database, but web files on the file system.

Defining the Storage Rule

In the *default* rule below, all files are stored in the database, while the *filesInWeb* rule stores the vault files in the database and the web files on the file system.

Example:

```
@ResultSet StorageRules
4
StorageRule
StorageType
IsWeblessStore
RenditionsOnFileSystem
default
JdbcStorage
filesInWeb
JdbcStorage


web
@end@
```

Defining the Path Construction

```
@ResultSet PathConstruction
4
FileStore
PathExpression
AutoCreateLimit
IsWritable
StorageRule
vault
$#env.VaultDir$$dDocType$/$dDocAccount/$dID$$ExtensionSeparator$$dExtension$
6
true
default
weburl.file
$HttpWebRoot$groups/$dSecurityGroup$/$dDocAccount$/documents/$dDocType$/
    $dDocName$$RenditionSpecifier$$RevisionLabel$$ExtensionSeparator$
    $dWebExtension$
3
false
default
web
$#env.WeblayoutDir$groups/$dSecurityGroup$/$dDocAccount$/documents/
    $dDocType$/$dDocName$$RevisionLabel$$RenditionSpecifier$
    $ExtensionSeparator$$dWebExtension$
3
true
default
@end
```

Altered Path Construction and Algorithms

The previous examples have kept the file paths consistent with the standard configuration. For very large implementations, this can result in directory saturation and slow performance. The following examples aid in dispersing files over several storage options.

Using Partitioning

File Store Provider makes it easy to use partitions to create a sparser directory structure. By default, the *xPartitionId* metadata field is used and becomes a part of a content item revision's metadata information. It is recommended that this field is disabled on the Content Server user interface, instead letting the partition selection algorithm determine the partition to use. The partition selection algorithm looks at all the active partitions, and as a new content enters the system, the partitions are selected in order. Each partition has an entry in the PartitionList Table and can be declared active. The PartitionRoot is calculated from the xPartitionId, where the value is a look up key into the PartitionList table. If no xPartitionId is specified, the system finds the next available and active partition and uses this value for the location calculation. The xPartitionId is then stored as part of the content item's metadata.

To use the partition selection, define the vault storage class in the PathConstruction table as follows:

```
vault
$PartitionRoot$/$dDocType$/$dDocAccount$/$dID$$ExtensionSeparator$$dExtension$
6
true
```

Partitions can be deactivated using the Add/Edit Partition Page (page 3-14) at any time if a system administrator needs to close a partition to contribution, for example if maintenance is required on the storage device.

Limiting the Number Files in a Directory

Another way of dispersing files is to alter the path so that files get partitioned out by the dID of the content item. In the example below, the directories are limited to 10,000 files plus extra files for additional renditions.

If you path expression contains

```
$dID[-12:-10:0]/$dID[-10:-8:0]$/$dID[-8:-4:0]$
```

and dID is 1234567890, the result is 00/12/3456

Note the dID[-12:-10:0] in the path expression. This is interpreted as follows:

- Get the characters starting at 12 back from the end of the string until you get the character 10 back from the end of the string.
- Pad the resulting string to length 2, which 12-10, with 0 characters.

Appendix

A

THIRD PARTY LICENSES

OVERVIEW

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

APACHE SOFTWARE LICENSE

W3C® SOFTWARE NOTICE AND LICENSE

ZLIB LICENSE

GENERAL BSD LICENSE

GENERAL MIT LICENSE

UNICODE LICENSE

MISCELLANEOUS ATTRIBUTIONS

Index