Content Server Full-Text Indexing System Installation and Administration Guide

Version 5.3 SP5
May 2007

Table of Contents

List of Figures

List of Tables

Preface

Purpose of the manual

This manual contains information and instructions you need to install, upgrade, and maintain the full-text indexing system used with EMC Documentum Content Server. It describes decisions you must make and requirements that must be met before you install the full-text indexing software. It also provides step-by-step instructions for installing and upgrading the software in several different configurations.

Intended audience

This manual is intended for the person installing Content Server and the full-text indexing software. Typically, a system administrator installs the software.

Revision history

The following revisions have been made to this document:

Revision history

Date	Description
May 2007	Initial publication

Acknowledgements

This product includes software developed by the Apache Software Foundation (www.apache.org). It installs Apache Tomcat.

Chapter 1

Introduction to Full-Text Indexing

This chapter presents a high-level overview of full-text indexing in Documentum deployments. It describes the components involved in indexing and provides a summary of the indexing process. The topics in this chapter are:

Benefits of an index

A Documentum repository can contain millions of objects. Locating objects containing particular values is a major challenge, whether you are trying to find a phrase, a date, or a name. Carefully structuring the folders in the repository can make the task somewhat easier by storing related objects together. DQL queries can rapidly locate objects whose metadata contains the values for which you are searching. Finding important information in a repository can still be like finding a single fish in a huge ocean.

A full-text index is one solution to the problem of rapidly locating the information you need in a repository. Full-text indexing is a process that indexes all content files and properties in a repository, creating an index that can be searched rapidly to retrieve objects whose properties or associated content files contain the values for which you are searching.

DQL, the Content Server query language, supports querying against the index. Queries can be executed against the index only or against the index and the metadata tables. For

information about the types of queries supported by DQL, refer to the *Search Development Guide*.

Content Server supports full-text indexing by default. However, Content Server itself does not create or maintain the full-text index. You must install the full-text indexing software components, which create and maintain the index. Full-text indexing components, page 14, contains a description of these components. This manual (*Content Server Full-Text Indexing System Installation and Administration Guide*) contains instructions for installing or upgrading the full-text indexing software.

What is indexed

A full-text index is an index on the properties and content files associated with objects of SysObjects and SysObject subtypes and, optionally, lightweight objects. Searching the index allows the rapid retrieval of objects whose properties or associated content files contain the values for which you are searching.

All properties of SysObject and SysObject subtype objects are indexed automatically. That cannot be turned off unless you turn off all indexing.

If any indexed object has an associated content file, the content file is also indexed if the a_full_text property of the object is set to TRUE and the format of the content file is indexable. If the a_full_text property is set to FALSE, the content file is not indexed. Content files in all storage areas are indexed.

If you install the full-text system with grammatical normalization enabled, then the index will contain normalized entries for terms within the documents also. For information about grammatical normalization and how it affects searches, refer to the *Search Development Guide*.

Phonetic searching is not supported.

Full-text indexing components

Full-text indexing in a Documentum repository is controlled by three software components:

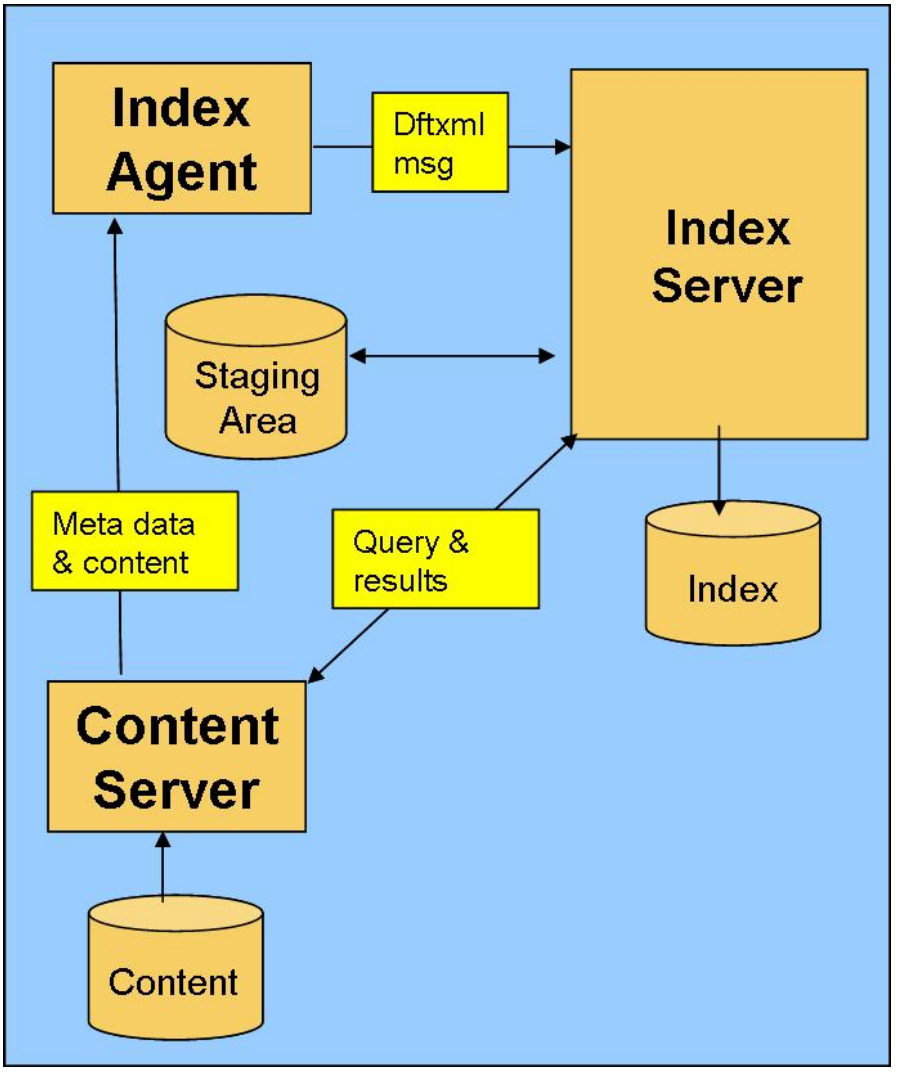
- Content Server, which manages the objects in a repository, generates the events that trigger full-text indexing operations, queries the full-text indexes, and returns query results to client applications
- The index agent, which exports documents from a repository and prepares them for indexing

- The index server, which is a third-party server product that creates and maintains the full-text index for a repository. The index server also receives full-text queries from Content Server and responds to those queries.

  The index server's operations are processor- and memory-intensive, and it is therefore recommended that you install the index server on a host other than the Content Server host. Multiple index servers may be installed on a single UNIX or Linux host, provided the directories, ports, and environment variables for each index server are configured properly. Only one index server can be installed on a Windows host.

Figure 1–1, page 16, illustrates the relationships among the Content Server, index agent, and index server. For indexing, the Content Server sends metadata and content for indexing to the index agent, which in turn packages that information as DFTXML and sends it to the index server. The index server processes the information and updates the index. For queries, Content Server sends the query to the index server, and the index server returns the results to Content Server. (The staging area is not a software component, but the directory used by the index agent and the index server. The index agent places content there for the index server to pick up to indexing.)

Figure 1-1. Full-text indexing components



About the full-text indexing process

The full-text index is updated on a continuous basis, provided that all of the software components are running. No special administrative tasks must be performed to ensure that the index is updated and current.

The indexing process begins when an object is added to the repository, removed from the repository, or updated in a way that might affect the full-text index. The process in response to the changed object is:

1. Content Server generates a queue item and adds it to the work queue for the index agent.
2. The index agent acquires the queue item from its work queue.
3. The index agent retrieves the object associated with the queue item from the repository and creates a DFTXML representation of the object. DFTXML is an XML format that contains the object's properties and the location of the object's content file, if any.

   If you are using distributed content and a content file is located at a remote component of the distributed store, the content file is copied to the distributed store component located at the primary site.
4. The index agent sends the DFTXML representation of the object to the index server.
5. The index server retrieves the content file associated with the object from the repository, if it has one, and creates its own representation of the content in the file and properties of the object. This internal representation is called FIXML.
6. The index server notifies the index agent that the FIXML is created and the object will be indexed, and the index agent destroys the queue item for the object.
7. The index server indexes the content file and its properties, adding its information to the full-text index.

   The object is now searchable, but note that the index server does not provide any indication that an object is searchable.

The indexing process is not destructive to existing content or attributes in the repository. The content files and object attributes are read, but not modified, during the indexing process.

Large file constraint

The full-text indexing system rejects for indexing any FIXML file larger than 10MB If that file is 10MB or larger, neither the content nor the properties of the object represented by the rejected FIXML file are indexed.

What languages can be indexed

Content files and properties in all supported languages are indexed by default. All standard Unicode character sets are supported. No special configuration is necessary.

Two right-to-left languages are supported for full-text indexing with certain limitations: Hebrew and Arabic. Other right-to-left languages cannot be indexed. For non-binary formats, only logical text representation is supported; visual text representation is not supported. For those binary formats listed below that support right-to-left text in the native format, support is provided for indexing Hebrew and Arabic text, with the exception that PDF files cannot be indexed.

Note: Grammatical normalization is available only for a subset of the supported languages. Grammatical normalization, page 19 briefly describes grammatical normalization. For a list of languages for which grammatical normalization may be used, refer to Whether to use grammatical normalization , page 43. The *Search Development Guide* describes the implementation of grammatical normalization in detail.

How particular characters are handled

The following Unicode characters are indexed and are searchable:

- Alphabetic characters
- Numeric characters
- Extender characters

  Extender characters extend the value or shape of a preceding alphabetic character. These are typically length and iteration marks.
- Custom characters enclosing Chinese, Japanese, and Korean letters and months

  These are derived from a number of custom character ranges that have bidirectional properties, falling in the 3200-32FF range. The specific character ranges are:

  — 3200-3243

  — 3260-327B

  — 327F-32B0

  — 32C0-32CB

  — 32D0-32FE

Other characters, including punctuation, accent, and diacritical marks, and characters such as | and #, are not indexed or searched. Such unsearchable characters are removed from the indexed text and treated as if they are blank spaces. The index server treats the following characters as white space:

```
!@#$%^_,.&;:()-+=<
```

When these characters appear in indexable content, they are replaced by white space. For example, when the email address MyName@company.com is indexed, it appears as

"MyName company com" in the index. The text is treated as three words. Documents returned by a search for MyName@company.com are treated as if they contain the words "MyName company com."

Note: Because the index treats that email address as three words, the document containing MyName@company.com would also be returned if the user conducted a phrase search on "MyName company com",

If a special character is included in a query, it is removed. For example, querying on `Richard+Dodd` would return a document containing the text `Richard=Dodd` because the + and = signs are both replaced by a blank space. If a search term includes an accent or diacritical mark, the search returns all matching words with or without the accent or diacritical mark.

Configuration options

There are several configuration options for the full-text system that provide enhanced control of the indexing or searching capabilities. You can configure:

- Whether or not to use grammatical normalization
- Specific rendition formats to index
- Processing of batched returns for non-FTDQL queries
- A thesaurus, to implement thesaurus searching

Grammatical normalization

Grammatical normalization ensures that all forms of a word are indexed and that a search for objects that have one form of a word also returns objects with other forms of the word.

When the full-text components are installed, you are offered the choice of enabling grammatical normalization. If you choose to enable it, grammatical normalization is enabled by default for English, Japanese, and Korean. You can choose additional languages and what combination of terms you wish to normalize. The default term choice is "nouns", meaning that the system will normalize all nouns found while indexing and specified in search queries. Using the default of normalizing nouns only is recommended.

For a full list of languages for which grammatical normalization may be enabled, refer to Whether to use grammatical normalization , page 43. The *Search Development Guide* describes the implementation of grammatical normalization in detail.

Rendition formats to index

Documents have content files in many formats. Some documents have primary content that is not indexable, but also have indexable renditions of that content. Other documents have indexable primary content and indexable renditions. By setting properties within the format objects for the various formats, you can direct Content Server to index all indexable renditions of a document or only preferred renditions. For instructions on setting format object properties to configure which renditions are indexed, refer to Configuring format objects to specify which renditions are indexed, page 96.

Processing of batched returns

When a query is run against both the full-text index and the database, the results, with certain exceptions, are processed by Content Server in batches. The processing is performed to remove duplicates if needed and to filter the results for security purposes. You can configure the size of these batches. For more information about the batches, their use and configuration, refer to Configuring batched returns for non-FTDQL queries, page 110.

Thesaurus usage

Adding a thesaurus to a full-text indexing system allows you to define which words are returned by a particular search. For example, suppose the thesaurus provides the following synonyms for “cat”: feline, polecat, cougar, and bobcat. When a user searches on the term ’cat’, the full-text engine will return documents containing feline, polecat, cougar, or bobcat also.

The thesaurus is implemented by adding a synonym file to the full-text installation. For instructions, refer toEnabling thesaurus searching, page 111.

Pre-installation planning decisions

Before you create a new repository in which full-text indexing is enabled, there are several decisions that you must make. For example, you must determine the appropriate deployment configuration for the index, including which hardware to use for the index agent and index server, and whether you want to use grammatical normalization and if so, which terms do you want to normalize.

For a detailed discussion of the indexing deployment models and guidelines for the choices you must make, refer to Chapter 2, Full-Text Indexing Deployment Models

Chapter 2

Full-Text Indexing Deployment Models

Full-text indexing is a resource-intensive process. The configuration of the major components of the indexing system (Content Server, index agent, and index server) has a significant impact on the performance of full-text searching. EMC Documentum supports different types of deployment models, each having different hardware and configuration requirements, and each providing a distinct set of benefits. Your business needs determine which full-text indexing deployment model is best suited to your implementation, as you make appropriate trade-offs between cost, system complexity, performance, and reliability.

This chapter discusses these full-text indexing deployment models and describes the considerations that help you determine which model is appropriate to your needs.

The models discussed in this chapter are:

- Basic deployments, page 24

  In this model, all three indexing components run on the same host or with a single index server serving a single repository
- Consolidated deployments, page 25

  In this model, a single index server serves multiple repositories.
- High-availability deployments, page 26

  This model has multiple index servers redundantly indexing a single repository to guard against system downtime.
- Multinode deployments, page 29

  In this model, subprocesses within the index server are installed across multiple machines to provide improved performance.

  Note: This model requires assistance from Documentum Professional Services to install.

Basic deployments

The basic indexing model consists of a single index agent and index server supporting a single repository. The index agent and index server may be installed on the Content Server host or on a different host. This model is also called a single-node deployment, because the index server is installed on one host computer.

Documentum supports the following two configurations for the full-text indexing components:

- Content Server, repository, index agent, and index server on a single host
- Content Server and repository on one host with the index agent and index server on a separate host

Each repository requires its own index agent. Consequently, if you have multiple repositories in a single Content Server installation, you must install a separate index agent for each repository. However, a single index server can serve multiple repositories. Deployments in which a single index server services multiple repositories are called consolidated deployment and are described in Consolidated deployments, page 25.

You can also install redundant indexing systems to support a single repository, in a high-availability configuration. For more information, refer to High-availability deployments, page 26.

Benefits

Basic deployments are easily installed and require little or no manual configuration. The installation procedure is fully supported.

The basic deployment is suitable for a development environment or a production repository with a low-to-medium volume of content created or modified.

A basic deployment may be more easily backed up and restored than a consolidated deployment, in which multiple repositories are indexed by a single index server, because the index data for different repositories cannot be separated out in a consolidated deployment.

In a generic content-management environment, it is expected that the ingestion rate is low. Therefore, a low latency requirement can be met by a basic deployment. However, note that the larger the index, the greater the save-to-search latency period.

If the index agent and index server are not on the Content Server host, indexing performance is improved by sharing the drive containing the repository's file stores with the indexing system host. Instructions for the manual configuration required are in Deciding whether to share the drives where content files are located, page 52.

Use considerations

Using a basic full-text indexing deployment is not recommended if:

- You have more than 20 million distinct objects that must be full-text indexed.

  Note: Having less than 20 million objects to index does not guarantee that a basic deployment is the correct configuration for your enterprise. Other data characteristics, such as the index size, affects the deployment decision also.
- The ingestion rate is expected to be high.
- The estimated size of the final full-text index is expected to be greater than 500 GB.

  Note: The hardware hosting the index must be sufficiently powered in relation to the size of the index. For example, an index of 500GB may not be successful on a basic deployment if its host is underpowered—indexing may take a long time, and querying may time out.
- The target repository is an archival repository.

  For more information on archival environments, refer to Archival repositories, page 39.
- If you have multiple repositories configured in an installation on a single host, and you want each repository to have its own index server, each repository will require a separate host machine for its index agent and index server. This requirement derives from the constraint that only one index server may reside on any particular host.

  An alternative in such cases is to use a consolidated deployment, in which all repositories use a single index server. Consolidated deployments are described in Consolidated deployments, page 25.

Consolidated deployments

In a consolidated deployment, a single index server provides search and indexing services to multiple repositories. The repositories may be in the same Content Server installation, in different installations on UNIX hosts or on different hosts. However, all repositories must be the same Content Server version

Consolidated deployments are easily installed, by configuring an index agent for each repository, each of which directs data to a single index server.

Benefits

Consolidated deployments require little or no manual configuration. Consolidated deployments reduce overhead by serving multiple repositories.

Consolidated deployments are suitable for a development environment or for production repositories with a low to medium volume of content created or modified.

Consolidated deployments are supported in a high-availability configuration.

If the index agent and index server are not on the Content Server host, indexing performance is improved by sharing the drive containing the repositories' file stores with the indexing system host. Instructions for the manual configuration required to share drives are in Deciding whether to share the drives where content files are located, page 52.

Use considerations

The index data for each repository in a consolidated deployment cannot be separated and cannot be discretely backed up or restored. If there is a business or capacity need to separately index or reindex one or more of the repositories, the index must be deleted and each repository reindexed.

If the total amount of indexed data begins to exceed the capacity of the host, migrating to one or more larger systems may be required.

A consolidated system that is configured as a single-node system has the same total volume and size constraints as a basic deployment configuration.

A consolidated system that is configured as a multinode system has the same total volume and size constraints as a single-repository on a multinode system.

High-availability deployments

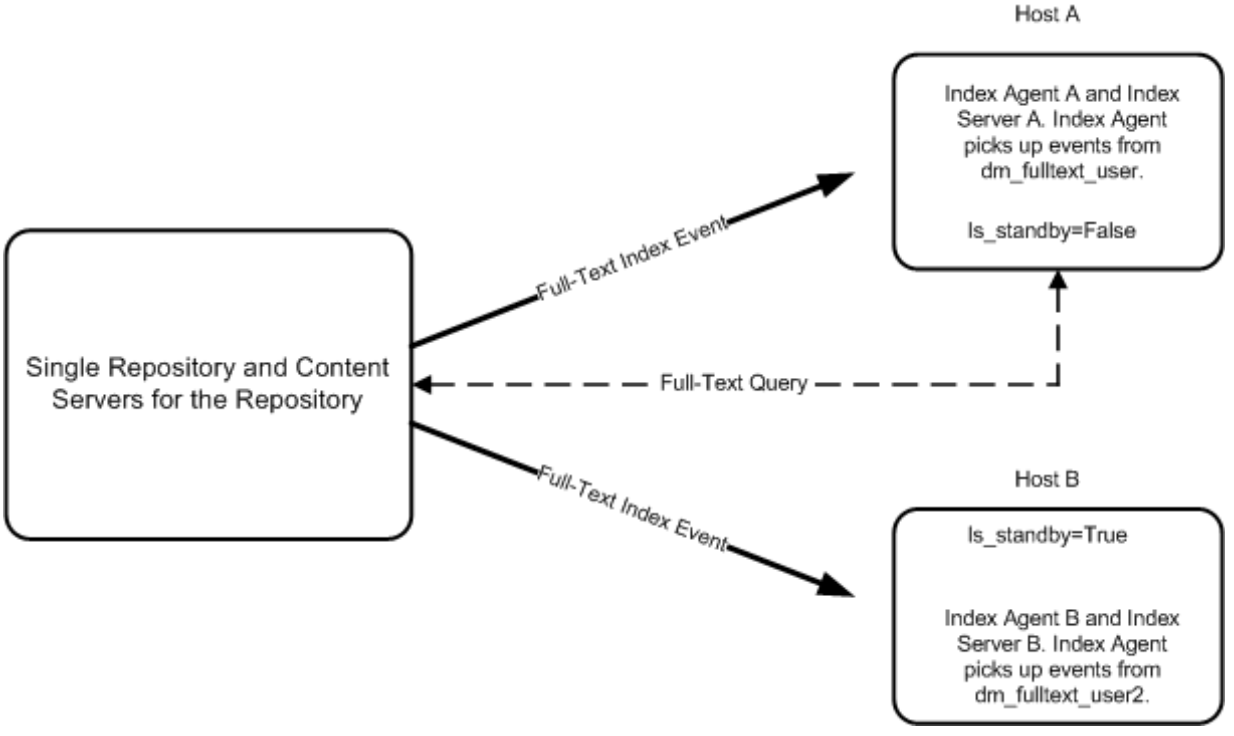
A high-availability deployment is a deployment in which two separate, fully-redundant indexes are created by running two or more indexing systems against a particular repository. If one of the indexing systems fails and the others continue to run, all search and indexing operations continue on the surviving systems.

High-availability deployments are supported in combination with consolidated deployments and multinode deployments.

In a high-availability configuration, separate instances of the indexing software are installed on two hosts (for example, host A and host B). Duplicate queue items

are generated for each indexable event. One queue item per event is queued to dm_fulltext_index_user and processed by the index agent on host A. The other queue item for each event is queued to dm_fulltext_index_user_01 (shown as dm_fulltext_user2 in the figure) and processed by the index agent on host B. The index servers host A and host B maintain separate, redundant indexes. Figure 2–1, page 27, illustrates this configuration.

Figure 2-1. Full-text high-availability configuration



The index on host A is considered the default index and the index on host B is considered the standby index. All full-text queries are directed to the index server on Host A.

If the indexing software on host A or host B fails, or if one of the hosts fails, the indexing software on the other host continues to process queue items and update the index. Indexing operations for the repository continue automatically on the remaining system. When the host or software that failed is again running, the index agent on that host acquires and processes any queue items that accumulated while the system was down.

If host A fails or if the indexing software on host A fails, querying must be manually switched to the index server and index on host B by making the standby index the default index. Refer to If a node fails in a high-availability configuration, page 118, for full details.

Note:

On the high-availability configuration and the load balancer — If a load balancer is used, there is no need to designate one index as a standby index. Additionally, with a load balancer, queries are directed automatically to either repository. For more information about the load balancer, refer to the whitepaper called *Full-Text High Availability Deployment*.

In a high-availability configuration, there is no guarantee that the indexes on each host are identical at a particular point in time. The index agents serving each index may acquire queue items at different rates, or network traffic may affect the speed of processing by an index agent or index server. Therefore, a query may return different results depending on the state of the index and which index server responds to the query.

High-availability configuration is supported with a consolidated configuration.

High-availability configuration is not supported with Microsoft Windows Cluster Services or in other clustering environments.

To create a high-availability deployment, several actions are required:

- An additional index queue user, dm_fulltext_index_user*N* is created for each *N*th index agent/index server pair.

  For example, when two indexing systems are running, the two queue users are dm_fulltext_index_user and dm_fulltext_index_user2
- The queue_user element in the indexagent.xml file in the additional index agent installation is modified to find the additional queue user.

Benefits

A high-availability deployment:

- Increases query availability, because some support is provided for failover.
- Provides redundancy if a host fails.

Installing a standalone high-availability deployment or a high-availability deployment with a consolidated deployment is supported. Documentum Administrator 5.3 SP2 or later provides tools for managing multiple index queues and for stopping and starting multiple index agents and index servers.

Installing a high-availability deployment in conjunction with a multinode deployment requires Documentum Professional Services for the multinode installation.

Use considerations

High-availability configurations are not supported on Microsoft Cluster Services. Some manual configuration is presently required to fail over querying if one of the hosts or one of the indexing installations in a high-availability deployment fails.

A high-availability deployment also requires multiple computers.

Multinode deployments

A multinode deployment is a full-text indexing configuration in which the index server is installed across multiple hosts and all of its subprocesses work together.

Typically, the index server's administrative processes are installed on one node, while document processors, indexers, and search servers are installed on as many nodes as are required by the anticipated size of the repository and index and the required throughput. The index data is spread out over the nodes in the installation. (In other words, each node contains unique index data.)

A complete multinode deployment is referred to as an index server instance. Each node on which a document processor, indexer, and search server is installed is referred to as a search instance. The Content Distributor on the administrative node determines on which node each document will be indexed and routes the documents to the correct document processor. The document processors, which each have the ability to communicate with all indexers in the installation, then route the FIXML representation of the document to the correct indexer.

Queries are processed in parallel by the index server instance, increasing querying efficiency. The QR Server on the administrative node issues queries to all search servers, then collects results and returns them to the Content Server. Additional nodes can easily be added to a multinode deployment, increasing both indexing and querying capacity.

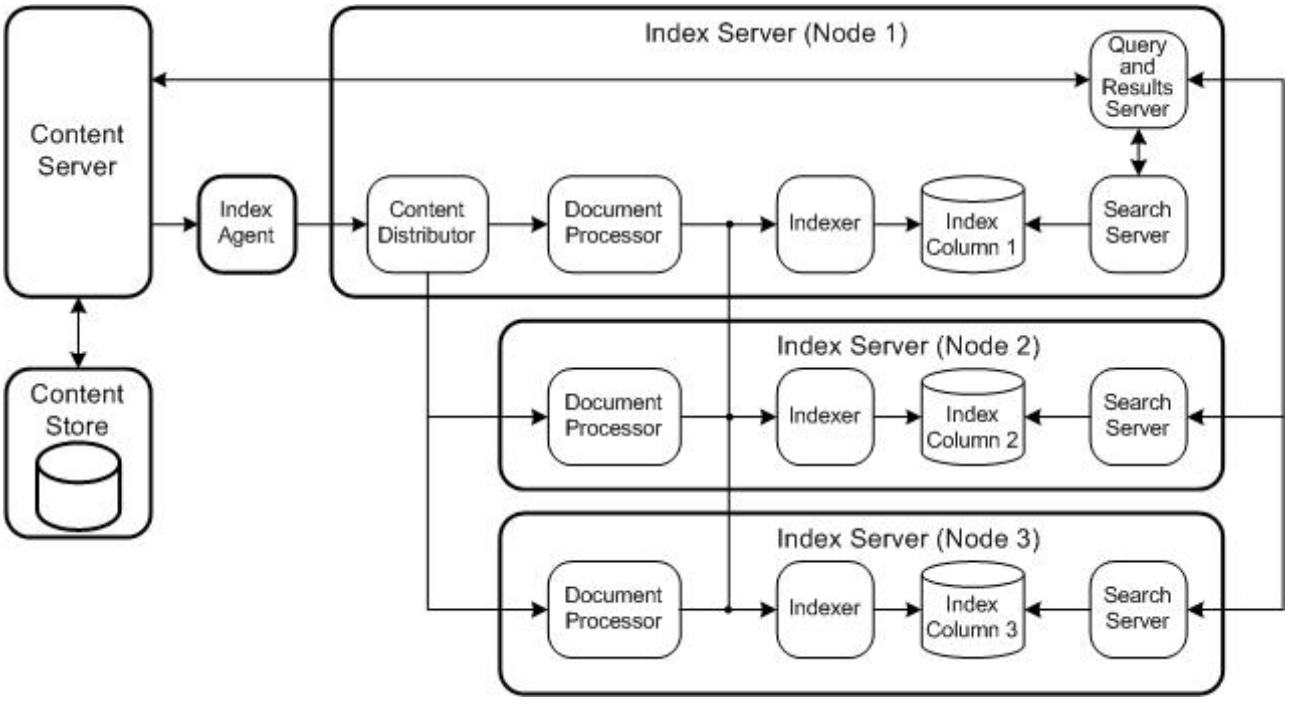
While multinode deployments are supported at runtime, you must use EMC Documentum Professional Services to design and install any multinode configuration.

Figure 2–2, page 30, illustrates a three-node indexing deployment. Node 1 hosts the administrative processes, including the content distributor and query and results server (QR server). Each node hosts a document processor, indexer, and search server as well as an index column.

The index agent passes DFTXML to the content distributor, which communicates with all document processors. The content distributor routes the document for processing. Each document processor communicates with all indexers. When the document processor concludes its work on the document, it passes the document to the correct indexer. The indexer updates the index on the indexer's node.

When Content Server sends a query to the QR server, the QR server routes the query to the search servers on all nodes. The search servers send query results to the QR server, which combines the results and returns the results to the Content Server.

Figure 2-2. Multinode configuration with three nodes



Benefits

Multinode deployments are best used where large volumes of data must be indexed and searched, high performance is required, and the index is expected to be 250–500GB or greater. Use the guidelines and information in the *Sizing Documentum 5.3 Full Text* whitepaper to determine whether you require a multinode configuration.

Use considerations

A multinode deployment requires substantially more advance planning and analysis than other deployment configurations. Installing a multinode configuration requires Documentum Professional Services. The proposed configuration must be submitted

to EMC Documentum for approval. Implementation cycles are longer and resource requirements are greater, in terms of the number of computers, disk space, and memory.

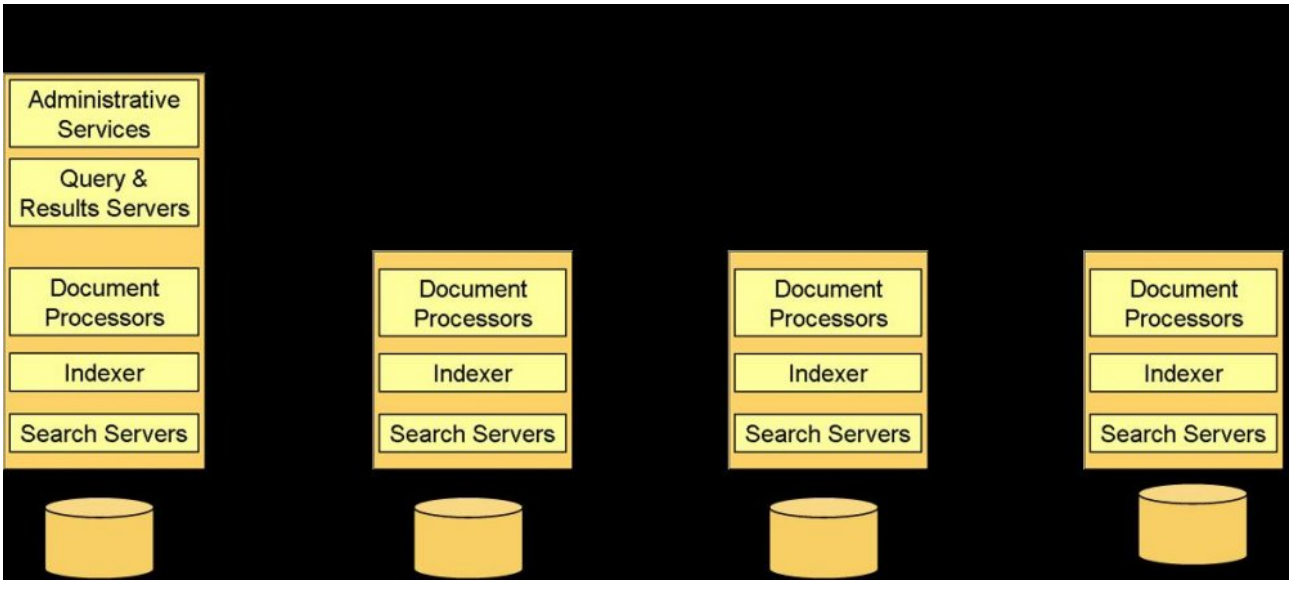
Supported multinode deployments

Documentum supports two models of multinode deployment. They are described in the following sections.

Basic multinode model

In this configuration, the administrative node includes a content distributor process that determines where incoming objects will be placed among the nodes. The content distributor routes documents to the docprocessors on each node in a predetermined order. Objects are typically directed sequentially across all nodes. For example, if there are four nodes, the first object is directed to the first node, the second to the second node, and so on, with the fifth object directed to the first node. The docprocessors route FIXML to the correct indexer process. All nodes taken together constitute the index server instance, while the nodes running only a document processor, indexer, and search server are search instances. Figure 2–3, page 31, illustrates this configuration.

Figure 2-3. Basic multinode configuration



This model is comparatively easy to deploy, with full installer support and the XML configuration files already set up. (Entries must be added to the XML configuration files for each node.) The model provides increased indexing throughput and a shorter latency period than a basic installation.

However, the basic multinode model has some considerations to keep in mind:

- All nodes are live and all backups must be full backups.
- To add a node after the initial deployment, the index must be rebuilt from the intermediate FIXML format in order to ensure that the data is balanced across nodes.

  Rebuilding the index also ensures that a document updated after the addition is stored in the same node as its previous version. Otherwise, if the document is routed to a different node, it is treated as a new document and a query may return duplicate rows for that document.

Multinode configuration with index routing

Index routing is a means of directing index data to specific nodes based on parameters you define in the repository and on the index server host.

Setting up a multinode configuration to include index routing provides several advantages:

- Control of where index data is stored.
- Easier to add additional nodes
- Allows older parts of an index to be less resource-intensive than current parts
- Nodes may be closed off by suspending indexing.

  Such nodes are backed up when they are closed off, but not after, because they are not updated.
- Only live nodes need to be backed up.

Note: If you change the routing, you must rebuild the index. If you do not, any documents that are updated after the routing change may be stored in a different column and treated as a new document. This can lead to duplicate rows returned for those documents by a query.

In archiving and other high-volume indexing scenarios, it can be beneficial to direct particular types of documents or documents produced during a discrete time period to a particular storage area, then map that storage area to a particular index collection and column in a multinode configuration. This routes documents to the particular index collection and column.

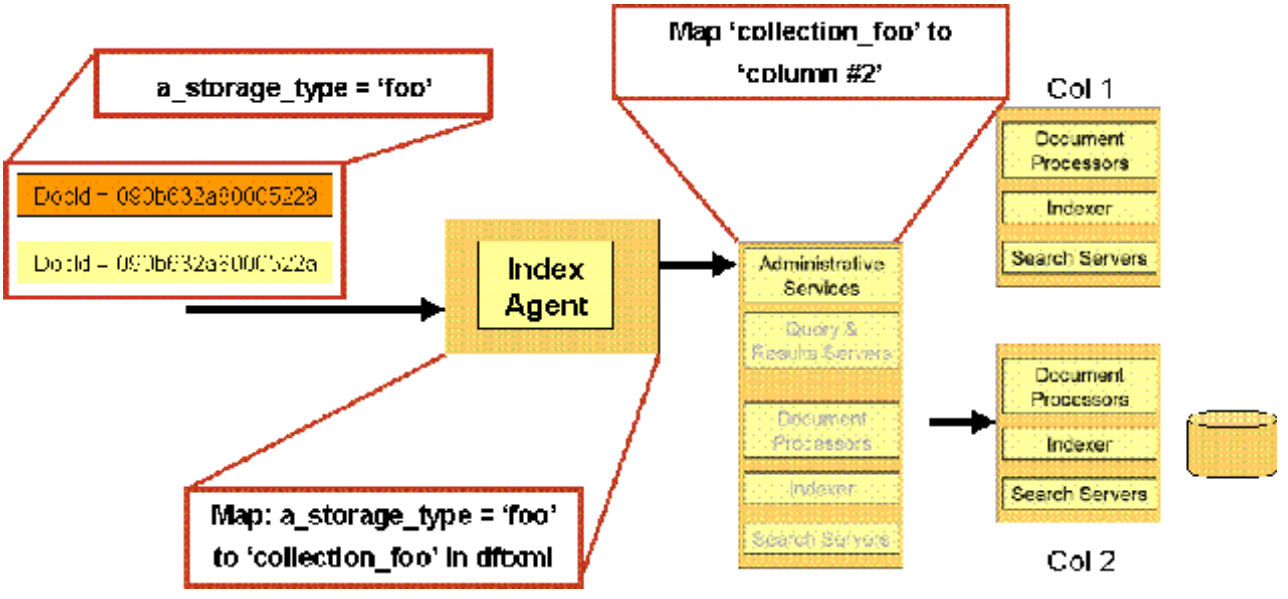
For example, a business may elect to store PDF documents in filestore_01, Word documents in filestore_02, and Excel documents in filestore_03. (File stores are the most common type of storage area in a Documentum repository.) To accomplish this,

the a_storage_type attribute of the SysObject is set on checkin for documents in each format. Documents are then stored in the correct file store by format. Another common archiving scenario is to direct documents to a particular storage area based on the time period during which the document enters the repository. File stores and index nodes are added as needed. For example, filestore_01 might contain documents checked in from January 1, 2006 through June 30, 2006 and filestore_02 might contain documents checked in from July 1 through December 31, 2006. A particular node can become inactive for indexing purposes once the end date is passed and documents begin to be routed to a new file store.

The index agent is then configured in the indexagent.xml file to direct objects to a particular logical collection at index time. Multiple file stores may be directed to the same collection. The index server is configured with the routing.cfg file to map a particular logical collection to one or more columns on one or more hosts. The routing.cfg file maps a collection to a column. An entry in the NodeConf.xml file on the administrative node notifies the status server of the requested physical partition of the logical collection.

Figure 2–4, page 33, illustrates the functioning of multinode configurations with index routing.

Figure 2-4. Multinode configuration with index routing



For instructions on configuring index routing, refer to Configuring index routing, page 100.

High-availability multinode

A basic high-availability deployment is described in High-availability deployments, page 26. High-availability deployment may be combined with multinode deployments to provide redundancy. In such a deployment, two index agents process duplicate queue items. If you are using a load balancer, it can interpret and respond to error messages from the index servers, to ensure that index information is not passed to a multinode installation that has failed. For information about the load balancer feature, refer to the whitepaper *Full-Text High Availability Deployment*.

Unsupported multinode configurations

The following multinode configurations are not supported:

- Multiple index server search instances on the same host, also known as multiple columns on the same host
- Multiple search rows

  Multiple rows have limited utility for high availability. The index server's administrative services cannot be made highly available within a single instance of the index server. If a particular search row indexer fails, the duplicate indexer also fails.

The configurations described in Supported multinode deployments, page 31 are the only configurations supported by EMC Documentum.

If these models do not meet your requirements

If the full-text indexing models described in this chapter do not meet the requirements of your installation, EMC Documentum Reporting Services may meet your needs. EMC Documentum Reporting Services is a product designed for particular types of installations, characterized by the following:

- Suitable for managing fixed content reports

  These are reports generated by other computer systems, usually in formats such as PDF, TIFF, and text. The content files are opened, and data is pulled from fixed format fields. This data is then loaded into a relational database.
- Not suitable for non-archival environments
- Content files contain data in fixed fields
- High ingestion volume
- Low query rates

- Highly selective queries, such as lookups on customer IDs
- Content not modified after intake

Chapter 3

Planning considerations

This chapter discusses the considerations that determine which of the deployment models you will choose to use for the full-text indexing system. The topics in this chapter are:

- Planning overview, page 37
- Determining the configuration, page 38
- Whether to use grammatical normalization , page 43

For information about the deployment configurations supported for full-text indexing installations, refer to Chapter 2, Full-Text Indexing Deployment Models

Planning overview

The following questions must be answered before you install a full-text indexing system for a repository:

- Which deployment model to use

  The deployment model you choose will depend on the answers to the following questions:

  — What is the purpose of the repository

  Purpose of the repository, page 38, discusses this consideration.

  — How many documents will be indexed

  — Number of documents to be indexed, page 41 discusses this consideration.

  How large are the documents to be indexed and how much indexable content do they contain

  Size of documents and amount of indexable content, page 41, discusses this consideration.

  — What file formats are to be indexed

  Content file formats to be indexed, page 42, discusses this consideration.

  - What quantity of metadata (property values) will be indexed

    Quantity of metadata to be indexed, page 42, discusses this consideration.

  - What are the business requirements regarding latency

    Indexing latency requirements, page 43, discusses this consideration.

- Which hardware to use for the index agent and index server

  Sizing the full-text indexing installation appropriately, based on the estimated size of the index and the chosen deployment model, is very important. An installation that is installed on underpowered machines or not appropriately configured can result in poor performance or query timeouts for users. System sizing, page 45, provides some guidance. Additionally, EMC Documentum provides a sizing spreadsheet that can help you determine the appropriate machines and sizing for your installation. This spreadsheet is available for downloading through Powerlink.

  It is recommended that you use a host other than the Content Server host for the index server.

  The index agent and the index server may be installed on a different supported operating system from the operating system on which Content Server is installed. The index agent and index server must be installed on the same host.

- Whether to mount or share the drives where the content files are located with the index server

  Refer to Deciding whether to share the drives where content files are located, page 52, for more information.

- Whether to use grammatical normalization

  Whether to use grammatical normalization , page 43, discusses grammatical normalization.

Determining the configuration

Use the guidelines in this section to choose a deployment configuration.

Purpose of the repository

Typically, a repository is either used for storing documents that are accessed regularly, to support on-going daily business processes or for storing archived documents that are not accessed on a daily basis. This section discusses how these two types of uses affect decisions about the full-text indexing system.

On-going content management repository

An on-going content management deployment is used to support on-going business functions. The repository generally contains a small to moderate number of documents (less than 10 million) that change over time. These documents enter the repository at a fairly slow rate of speed and may be updated from time to time. The documents are created, edited, and deleted by individuals within the organization. The content files are in the mix of formats required by the organization's business needs, and may include such formats as Word, Excel, PDF, Visio, JPEG, and other graphic formats, AutoCad, text, XML, and other commonly-used formats.

In such a repository, the content files and metadata are updated and versioned on a regular basis. Individuals query the repository in order to locate particular documents on an ad-hoc basis.

The full-text indexing needs of such a repository may be met by a variety of configurations, depending on the size of the repository, the volume of new material in the repository, the latency requirements, and the volume of queries issued by users.

Archival repositories

An archival repository is used to store large volumes of unchanging data. Such repositories typically contain up to 100 million documents and require high throughput. The content files are in a limited number of formats (for example, TIFF, PDF, and text) and are typically generated by computers, scanners, and software applications rather than individuals. The data may consist of email, bank statements, credit card statements, and other fixed-format or fixed-field documents. The content files are rarely, if ever, modified.

An archival repository requires a full-text indexing solution that has the capacity to index large quantities of data at high speeds. The requirements for querying capacity is driven by particular applications. For example, legal discovery and data-mining applications may require a high capacity for full-text querying. Other applications may search metadata values only, not the content files.

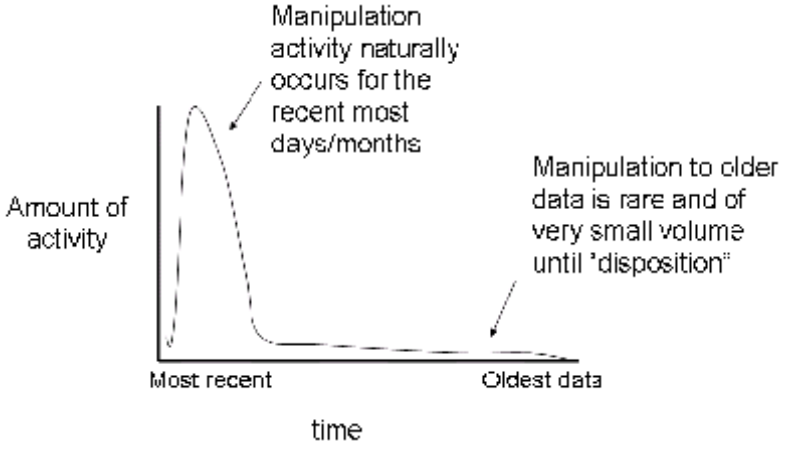
Considerations for an archival repository

Use the following information to help size and configure an archiving deployment.

Choosing CPU size and capacity

The data stored in an archival repository may be ingested rapidly or at a moderate rate. The quantity of data increases over time and the indexing system may need to maintain hundreds of millions (or even billions) of documents. But all cases, the data is stored for a long period of time, due to specific business needs or in response to regulatory or compliance requirements, and is most likely to be changed closer in time to when it is added to the repository than later, long after it was added. Older data may be purged from the system when the legally-mandated retention period has been exceeded. Figure 3–1, page 40, illustrates this principle.

Figure 3-1. Activity on archived documents over time



Define computer system resources (CPU, memory, I/O capacity) according to the required indexing throughput rather than according to the size of the index itself. The anticipated size of the index will determine the required disk space, however.

Multinode considerations

In planning the indexing configuration for an archival repository, another issue must be taken into consideration. When a multinode configuration is used, the index data is directed to different nodes, so that each node contains a mixture of old and new data. The mix changes over time, so that after four years, some three-quarters of the data will be more than one year old and thus least likely to change or be needed for searching.

Adding documents is a comparatively expensive process, and it is therefore advantageous to isolate older data. This can be done by using directed routing to route documents to specific collections and columns, localizing data by age to the collections. Directed routing, page 127, contains full details. Configuring directed routing is a

manual process requiring Documentum Professional Services. The process is described in Configuring index routing, page 100.

Number of documents to be indexed

The number of documents to be indexed in a particular period must be taken into consideration in deciding on both the index server configuration and the hardware you use.

The total number of documents to be indexed impacts the software in several ways. The most important is the potential to run into the per-process memory limits of the search processes that work on the largest of the partitions. These processes perform complex caching operations, and one of the caches grows in proportion to the size of the partition. At 10 million documents, this cache reaches 1/2 GB. At 20 million documents, the cache grows to 1 GB. The search server process is limited to 2 GB of virtual memory, which needs to be used for thread stacks and other caches, as well. Consequently, if the document load projections are between 10 and 20 million documents, then you should consider using a multinode configuration.

Document throughput may also make a multinode deployment desirable. Throughput is the rate at which new objects are added to the system or submitted for indexing. Higher throughput requirements result in higher processing costs. When installed on a dual-processor host, the index server is able to index about 60,000 documents per hour in a basic deployment (depending on the size of the documents and the hardware on which the system is installed). In a multinode configuration, the index server's throughput can be increased significantly. However, when a second node is added to the index server, the speed and capacity of the machine hosting the RDBMS database may need to be increased. For additional guidelines and information, refer to the *Full-Text Agent Throughput* whitepaper.

Size of documents and amount of indexable content

The size of an index is determined by the size of the largest documents indexed and the amount of indexable content in the documents. A very large file can contain a small amount of indexable content – text and date information – and a large amount of unindexable content, such as graphics.

Table 3–1, page 42, lists a sample of the figures used to size an index. The example is based on documents of a custom type in which most of the documents were associated with very small content files. More than 10 million documents are indexed, but 85% of the size of the index results from the 20 largest files, and very likely the index itself

would be significantly larger, up to twice the size, if the documents were typical sizes rather than artificially small sizes.

Table 3-1. Data characteristics of FIXML and index for 10 million documents

Characteristic	Sample value
Number of documents	10,000,000–plus
Number of files in the FIXML area	100,000
Total size of FIXML area	5.68 GB
Largest file in FIXML area	50 MB
Number of files in the index area	3,178
Total size of index area	60 GB
Largest file in the index area	14 GB
Total index size occupied by 20 largest files	51 GB, or 85% of the total index

If the documents to be indexed are very large, we recommend a multinode deployment so that the index itself is spread out over multiple hosts.

Content file formats to be indexed

Because the amount of indexable content in different file formats varies, the mix of content files in a particular repository influences the size of the resulting index. For example, graphic files generally do not contain indexable content, so that only the document metadata is indexed. XML files are essentially text files and contain a high percentage of indexable content. The index for a repository containing a high percentage of XML, text, or word processing files will typically be larger than the index for a repository of similar size containing primarily graphic files.

Quantity of metadata to be indexed

If the metadata associated with documents to be indexed includes large quantities of string data or a large number of custom attributes, the size of the index increases.

Indexing latency requirements

Indexing latency describes the period of time from when an object is saved in the repository to when the object is searchable. Your business may require a low-latency environment, in which an object becomes searchable as fast as possible. Typically, this is a requirement for production systems in which many searches are performed and many objects are created and edited. In an archiving scenario, large quantities of unchanging business data are stored, but rarely or never modified. In such an environment, a longer latency period may be acceptable.

Certain indexing configurations may reduce save-to-search latency. However, some strategies increase the resource requirements and risk in an indexing deployment.

If a business requires the shortest possible latency period and the content to be indexed is generic business formats, a multinode deployment provides a faster save-to-search time than a single-node deployment. If the content to be searched contains fixed fields rather than free text, a solution other than multinode full-text indexing may be more appropriate.

If a high-latency indexing environment is acceptable, some performance advantages may be gained by using batch processing. Indexing may be suspended, so that FIXML continues to be produced but new objects are not added to the index. When indexing is resumed, the new FIXML is processed and the new objects are added to the index.

Whether to use grammatical normalization

Grammatical normalization is a process that the full-text indexing system performs when indexing and when processing a query if the feature is enabled. The process ensures that queries return not only exact matches for terms, but also grammatical matches. For example, a query on 'cat' would return objects containing not only 'cat' also those containing 'cats'. (The *Search Development Guide* contains a full description of how grammatical normalization behaves during indexing and querying.)

Grammatical normalization is enabled during index server installation. Enabling grammatical normalization loads special dictionaries into the index server host's memory. By default, grammatical normalization is enabled and the dictionaries for Japanese, Korean, and English are loaded. You can choose to load additional dictionaries to support other languages. However, the memory requirements for the additional dictionaries may result in lower search and indexing performance. The additional supported languages for grammatical normalization are:

- German
- English
- Spanish

- French
- Hungarian
- Italian
- Swedish
- Norwegian
- Polish
- Portuguese
- Russian

There is no dictionary for Chinese because that language is not an inflected language and therefore grammatical normalization is not an operation that can be performed on Chinese text.

You can also choose the combinations of parts of speech you want to be normalized. The following combinations are offered:

- Nouns
- Nouns and adjectives
- Nouns, adjectives, and verbs
- Nouns and verbs

The default, and recommended choice, is to normalize only nouns.

Enabling grammatical normalization increases memory use by the indexing process. You cannot change the grammatical normalization option after installation. To change the grammatical normalization option, you must uninstall the indexing software and delete the index, then reinstall the software and index the entire repository again.

Chapter 4

Preparing to Install Full-Text Indexing

Use the information in this chapter to prepare your installation for the full-text indexing component installation. This chapter contains the following information:

System sizing

System sizing is the process of determining what hardware, software, and network configurations will provide the best performance for users at the lowest cost to the enterprise. Another term for system sizing is capacity planning.

Full-text indexing is both CPU- and disk-intensive. Indexing a repository may require disk space ranging from 3 to 10 times the size of the finished index. For this reason, it is critical that your system have sufficient CPU and disk space capacity.

EMC Documentum provides tools and guides for system sizing. The tools include information on full-text indexing configurations, including determining the appropriate number of nodes for the capacity you require. Documentum Professional Services can assist you in reviewing your needs and completing the spreadsheet.

A whitepaper, called *Sizing Documentum 5.3 Full Text*, is available that contains information to use in estimating the system resources required for the hardware on which the full-text indexing software will be installed. The whitepaper is available on the Technical Support Web site on Powerlink. The following general considerations must be taken into account:

- The number of CPUs on the host chosen for the indexing software
- The type of disk system chosen:
  - NAS devices are not supported for storing the indexes or FIXML.

  - SAN devices and disk arrays are recommended for storing the indexes and FIXML.
    - Allows use of a variety of disk systems
  - It is strongly recommended that you use a disk array in production environments, rather than a single disk.

    A single disk creates a bottleneck for indexing.
- Use gigabit Ethernet, not 100BaseT, in production environments
- It is recommended that you use high-capacity SAN fibre interconnects to the disk arrays.

Memory requirements for index server

The index server requires 2 GB of RAM available after taking into consideration all other RAM utilization requirements on the index server host.

Disk space requirements for indexing and installation

For information and guidelines about determining the disk space requirements for an indexing installation, refer to the *Sizing Documentum 5.3 Full Text* whitepaper.

Full-text indexing in a distributed content environment

If you are using distributed content, all content is copied to the primary content store for full-text indexing. Ensure that there is sufficient space in the primary store for all content that may be copied for full-text indexing.

Device types on which the full-text index and content files may be stored

Documentum content files may reside on a variety of storage devices, including (but not limited to) Centera stores, direct-attached disks, NAS, and SAN devices, provided that the storage device functions as a file-system directory and does not require a password for Content Server or indexing software access. NFS mounts are fully supported and

encouraged for access to original content files. However, they are not supported for storage of the FIXML files or the index.

Documentum recommends that you store the full-text index on direct-attached disks or SAN devices. Storing the index on an NFS-mounted device affects document processing performance and the amount of time it takes for a document to be present in the index and available for querying. It is strongly recommended that you do not store the index on an NFS-mounted device.

On Windows hosts, do not store the index on a mapped drive. Use a UNC path only.

Storing the indexes, log files, and FIXML on a NAS device is not supported.

Constraint on SAN devices

When SAN remote mirrors are used, only one index server can be running on the data (both indexing and querying at the same time). Running two instances of the index server from two different sites, one local and one remote, on the same index is not supported.

Host requirements

The following sections contain host requirements for installing the full-text indexing software.

Host names

The host where the index server and index agent are installed must be identified by a fully-qualified domain name. For example, the host name isolde.documentum.com is acceptable, but an IP address (for example, 172.04.8.275) is not acceptable.

Which ports to use for the index agent

The index agent runs in the Apache Tomcat servlet container. When an index agent instance is configured, you must designate two ports for the index agent and Tomcat to use. The default ports for the first index agent on a host are 9081 and 9008. If the index agent is on the Content Server host, ensure that the ports are not the ports used for the Tomcat instance in which the Java method server and ACS server run.

Which ports to use for the index server

The index server requires a contiguous range of 4000 (four thousand) free ports. You must designate which ports to use during installation. The default range is from 13000 to 17000.

Index server operating system and host

The index server must be installed on a supported operating system. It is strongly recommended that you use a host on which a clean installation of the operating system has been performed.

Install the index server on a disk partition separate from the system partition and larger than the system partition.

VMware

Do not install the index server on VMware.

Third-party software on the index server host

The following restrictions apply to the index server host:

- Do not run network security scanning software on a host where the index server is installed.

  Network security scanners may lock index server processes, which can create intermittent search and indexing failures.
- Do not run backup utilities while the index server is running.

  Backup utilities may lock indexing processes.
- Do not run antivirus software on the %FASTSEARCH% directory, where the index server and indexes are stored.

  Antivirus software interferes with index server startup and proper functioning. Antivirus software may quarantine log and other frequently-changed files.

It is strongly recommended that you test any third-party monitoring tools on a development system before the tools are deployed to a production system where the index server is installed.

Windows host requirements for the index server

The following restrictions apply to Windows hosts:

- Do not run the Windows Index System on the index server host.
- On 32-bit Windows hosts, do not set the /3GB option in the boot.ini file.
- Disable automatic Windows updates on index server hosts.
- Do not install the index server on a domain controller.

Note: If a 5.2.x repository is running on a Windows host and you are performing a pre-upgrade index migration, you must install the index agent and index server on a host other than the Content Server host. For more information, refer to Migrating the full-text indexing system, page 59.

Host time settings

Set the time zone on the host where the index server runs to GMT or UTC. On Windows hosts, uncheck **Automatically adjust clock for daylight saving changes**.

Ensuring correct network configuration

If you are installing the indexing software on a host other than the Content Server host, ensure that the DNS entries for the two machines are correct so that they are able to locate each other on the network.

To verify the DNS entries:

1. On the index server machine, look up the Content Server machine:

   `nslookup` *`FQDN_of_Content_Server_host`*

   where *FQDN_of_Content_Server_host* is the fully-qualified domain name (FQDN) of the Content Server host.

   This returns one or more IP addresses for the Content Server host.

2. Use the first IP address returned in step 1 for a reverse lookup:

   `nslookup` *`IP_address_returned`*

   The correct return value is the same FQDN you entered in step 1.

3. If the two nslookup commands do not return the correct values, update the DNS servers used by the two hosts to reflect the correct FQDNs.

4. If necessary, on Windows with more than one network card, update the host files to ensure that the correct IP address for each host is listed first.

5. If the nslookup commands succeeded and return the correct values, ping the index server host from the Content Server host to ensure it responds to the pin and to ensure that the IP address that responds to the ping is the IP address defined in the ftengine config object.

Index agent and index server installation account

The index agent and index server must be installed as the same user who installed Content Server (the Content Server installation owner). If you are installing the index agent and index server on a host other than the Content Server host, ensure that the user exists on that host. Refer to "Installation Owner Account" in the correct section for your operating system, in Chapter 3, "Preparing for Installation," of the *Content Server Installation Guide*, for more information on the installation owner account.

Environment variables on UNIX and Linux hosts

You must set the following environment variables in the installation owner's environment on UNIX and Linux hosts before installing the index agent and index server:

Table 4-1. Required environment variables

Environment Variable	Description	Required Values
DOCUMENTUM	The directory in which the indexing software is installed	Any directory in the installation owner's environment
DOCUMENTUM_ SHARED	The directory in which DFC is installed	Any directory in the installation owner's environment
LD_LIBRARY_PATH, SHLIB_PATH, or LIBPATH	The index server library location	$DOCUMENTUM/fulltext/ IndexServer/lib $DOCUMENTUM/fulltext/ fast40 $DOCUMENTUM_ SHARED/dfc $DOCUMENTUM_ SHARED/IndexAgents/ ftintegrity

Environment Variable	Description	Required Values
FASTSEARCH	Location of the index server	$DOCUMENTUM/fulltext/IndexServer
DISPLAY	Controls the display	localhost:0.0
LC_ALL		C
JAVA_HOME	The home directory for the Java installation on the host	Any directory in the installation owner's environment

Ensuring that the index server environment is correct on UNIX and Linux hosts

The index server installation includes a script that sets required environment variables for running the index server. The script is setupenv.sh or setupenv.csh, depending on the shell from which you run, and it is located in the *indexserver_install_dir*/bin directory. You can source this script to ensure that the environment variables are correct.

The deprecated DFC_DATA environment variable on UNIX hosts

The DFC_DATA environment variable was deprecated after the 5.1 Documentum release, but it is still used by Documentum installers for backward compatibility. If you are installing the indexing software on a UNIX host where older Documentum software required setting DFC_DATA, the installer uses the value of DFC_DATA to create the /config directory ($DFC_DATA/config). However, the startupIndexAgent.sh script expects to find the $DOCUMENTUM_SHARED variable set and expect the /config directory to be $DOCUMENTUM_SHARED/config.

If the /config directory is not $DOCUMENTUM_SHARED/config, edit the startupIndexAgent.sh script so that it points to the valid /config directory path on the index agent host. Replace these lines:

```
CLASSPATH=$DOCUMENTUM_SHARED/dctm.jar:$DOCUMENTUM_SHARED/config:
$DOCUMENTUM_SHARED/dfc/dfc.jar:$DOCUMENTUM_SHARED/dfc/dfcbase.jar:
$DOCUMENTUM_SHARED/dfc/log4j.jar
```

with:

```
CLASSPATH=$DOCUMENTUM_SHARED/dctm.jar:$DOCUMENTUM/config:
$DOCUMENTUM_SHARED/dfc/dfc.jar:$DOCUMENTUM_SHARED/dfc/dfcbase.jar:
$DOCUMENTUM_SHARED/dfc/log4j.jar
```

Installing the index server on Windows hosts

The Windows file cscript.exe is required for running the index server installer. Do not delete cscript.exe from a Windows host on which you are installing the index server.

Installing the index server on HP-UX

This section documents requirements specific to installation on the HP-UX platform.

Directory constraint

You cannot install the Index Server on HP-UX (B.11.23 U 9000/800) in a directory that contains an "_d" in the directory path.

Required parameters

The following parameter values are required for installing the index server on HP-UX systems:

- Set maxdsiz or data seg size at 2 GB (0x80000000)
- Enable Largefiles
- Set maxusers to 256 or higher
- Set max_thread_proc to 256 or higher
- Set maxfiles to 1024 or higher

Deciding whether to share the drives where content files are located

The index server requires access to the content files in a repository.

If the index server is installed on the Content Server host, it has direct access to the file store storage areas and you do not need to take any additional action.

If the index server is not installed on the Content Server host, the default behavior of the index agent is to retrieve a temporary copy of the file, store it in a temporary location, and pass that location to the index server. After indexing the file, the index server deletes the temporary copy.

For performance reasons it is recommended, but not required, that you mount or share the drive or drives where the repository's file store storage ares are located with the host where the index server is located. When the drives are shared or mounted, the index agent passes to the index server the direct path to a file that must be indexed. No staging area is required, and fewer I/O operations are required. You can share or mount the drive or drives so that the content files are read-only.

Note: Even when the file store storage area drives are mounted, XML content is retrieved using the temporary file method rather than the direct access method. In addition, content located in Centera stores, external stores, and encrypted file stores must be retrieved for indexing using the temporary file method.

Mount or share the drives before you install the indexing software. After installation, edit the indexagent.xml file to properly map the file stores for the index agent and use the index agent's administrative interface to indicate that the file stores are mapped. Modifying the indexagent.xml file to map file stores, page 71 contains instructions for editing the file.

In a distributed configuration, all content located in distributed stores is moved to the primary site, using the Surrogate Get method, and indexed at the primary site. You cannot map the remote components of a distributed store. Similarly, you cannot map an encrypted store or an external store.

You can share or mount the drive or drives so that the content files are read-only.

When you mount or share the drives, it is strongly recommended that you mount or share them so that the paths are logically identical on the Content Server host and on the index server host. On Windows hosts, use UNC paths. On UNIX, use NFS and, if necessary, symbolic links. If you must mount from a Windows platform to a UNIX or Linux platform, use third-party utilities for mounting or sharing the drives. The instructions for modifying the indexagent.xml file are different, depending on whether the paths are logically identical or not.

In a multinode configuration, if you do not mount or share the file store locations, it is recommended that you mount or share the temporary location where the index agent stores indexable content. The docprocessor processes, which open content files and extract indexable content, need access to the temporary location; share the location with all index server nodes that run docprocessor processes.

Refer to the documentation for your operating system for instructions on sharing or mounting drives.

In summary, here are the recommendations:

- If the indexable content includes XML content, or if some content is located in Centera (content-addressed stores), external stores, or encrypted file stores, mount or share the index agent's temporary content storage area.
- If the temporary file method is used, mount or share the index agent's temporary content storage area.
- If the direct access method is used, share or mount the public file store directories.

Chapter 5

Upgrading Full-Text Indexing Components

This chapter provides the instructions for upgrading the full-text indexing components.

Note: In a consolidated deployment, a single index server provides indexing services to multiple repositories. In any indexing configuration, the indexing software and Content Server(s) must have the same version number. Therefore, to upgrade a consolidated deployment, the indexing software and all repositories must be upgraded simultaneously.

This chapter contains the following topics:

Adding the full-text indexing system to a 5.3 repository that has none

Use the instructions in this section if you want to add the full-text indexing system to a 5.3 repository that currently does not have full-text indexing installed.

To add a full-text indexing system to a 5.3 repository:

1. Upgrade the repository from 5.3 to the current release.
2. Install the full-text indexing software.
3. Configure an index agent in migration mode.
4. Create the full-text index.
5. Use the Index Agent Admin Tool to change the index agent from migration mode to normal mode.

Upgrading an existing full-text system on a repository

Use the instructions in this section if you are upgrading from a 5.3 SP2 or 5.3 SP3 release that has not had the Full-text Hotfix of December 2006 installed.

If you previously installed the 5.3 SP2 or 5.3 SP3 Full-text Hotfix or previously upgraded to 5.3 SP4,, use the instructions in Upgrading from the December 2006 Full-text Hotfix or 5.3 SP4, page 57. If you are unsure whether the earlier Hotfix release has been installed, check the Content Server version number:

- On Windows, navigate to C:\Documentum\product\5.3\bin and enter:

  ```
  documentum.exe -version
  ```

- On UNIX, navigate to $DM_HOME/bin and enter:

  ```
  documentum -version
  ```

If you have installed the December 2006 Full-Text Hotfix, the version numbers will be:

- On 5.3 SP2: 5.3.0.229
- On 5.3 SP3: 5.3.0325

Caution: Do not use this procedure if the deployment is a multinode deployment. Contact Professional Services for instructions on upgrading a multinode deployment.

To upgrade the full-text indexing components:

1. Log in as the Content Server installation owner.

   Note: On Windows, this means that you must log in as the same user, in the same domain, as the user who installed Content Server and the full-text indexing software.

2. Stop the index agents and index server.

   If you did not stop these processes when you upgraded Content Server, stop them now. Make sure all index server processes are stopped.

3. Delete all index agent instances.

   Use the Index Agent Configuration Program to delete the index agents. Refer to Deleting an index agent, page 138, for instructions.

4. Uninstall the Index Agent Configuration Program.

   Refer to Deleting the index agent configuration program, page 139, for instructions

5. Uninstall the index server while preserving the index itself.

   The uninstaller will present a panel that asks if you wish to delete the existing index. The default is “no”, which preserves the index. Accept the default.

6. Manually delete the IndexServer directory.

Note: This step is only necessary if the index server is on a different host than Content Server. If the index server is on the same host as Content Server, uninstalling the index server will also remove the IndexServer directory.

This directory is found in the following location:

- On Windows: C:\Documentum\fulltext\IndexServer
- On UNIX: $DOCUMENTUM/fulltext/IndexServer

7. Manually delete the following directory:
   - On Windows: C:\Documentum\data\fulltext\index
   - On UNIX: $DOCUMENTUM/data/fulltext/index

Caution: Do NOT delete the fixml directory.

8. Install the index agent and index server.

   Follow the instructions in Installing the index server and the index agent configuration program, page 67 . Using the fulltextWinSuiteSetup.exe (fulltext*operatingsystem*SuiteSetup.bin) from the 5.3 SP4 release.

9. Shutdown and restart Content Server.
10. Start the index server.

    After you start the index server, it will recreate the index using the fixml files found in C:\Documentum\data\fulltext\index ($DOCUMENTUM/fulltext/IndexServer). To determine the status of the index server, check the all.log file for the entry: fnet: engine up. When that entry is present, the server has completed reindexing. Note that on a moderately powered machine and assuming there is fixml representing approximately 1 million documents to reindex, the process can take up to 10 hours.

    The all.log file is found in C:\Documentum\fulltext\IndexServer\var\log\all.log (Windows) or $DOCUMENTUM/fulltext/IndexServer/var/log/all.log (UNIX).

11. After the reindexing is complete, configure the index agents.

    Use the instructions in the Configuring the index agent, page 69.

Upgrading from the December 2006 Full-text Hotfix or 5.3 SP4

Use this procedure if you previously installed the December 2006 Full-text Hotfix release or 5.3 SP4. If you are unsure whether the December Hotfix release has been installed, check the Content Server version number:

- On Windows, navigate to C:\Documentum\product\5.3\bin and enter:

```
documentum.exe -version
```

- On UNIX, navigate to $DM_HOME/bin and enter:

```
documentum -version
```

If you have installed the December 2006 Full-Text Hotfix, the version numbers will be:

- On 5.3 SP2: 5.3.0.229
- On 5.3 SP3: 5.3.0325

Caution: Do not use this procedure if the deployment is a multinode deployment. Contact Professional Services for instructions on upgrading a multinode deployment.

To upgrade the full-text indexing components:

1. Log in as the Content Server installation owner.

   Note: On Windows, this means that you must log in as the same user, in the same domain, as the user who installed Content Server and the full-text indexing software.

2. Stop the index agents and index server.

   If you did not stop these processes when you upgraded Content Server, stop them now. Make sure all index server processes are stopped.

3. Delete all index agent instances.

   Use the Index Agent Configuration Program to delete the index agents. Refer to Deleting an index agent, page 138, for instructions.

4. Uninstall the Index Agent Configuration Program.

   Refer to Deleting the index agent configuration program, page 139, for instructions

5. Uninstall the index server while preserving the index itself.

   The uninstaller will present a panel that asks if you wish to delete the existing index. The default is “no”, which preserves the index. Accept the default.

6. Manually delete the IndexServer directory.

   Note: This step is only necessary if the index server is on a different host than Content Server. If the index server is on the same host as Content Server, uninstalling the index server will also remove the IndexServer directory.

   This directory is found in the following location:

   - On Windows: C:\Documentum\fulltext\IndexServer
   - On UNIX: $DOCUMENTUM/fulltext/IndexServer

7. Install the index agent and index server.

   Follow the instructions in Installing the index server and the index agent configuration program, page 67 . Using the fulltextWinSuiteSetup.exe (fulltext*operatingsystem*SuiteSetup.bin) from the 5.3 SP4 release.

8. Shutdown and restart Content Server.

9. Start the index server.
10. Configure the index agents.
    Use the instructions in the Configuring the index agent, page 69.

Upgrading a pre-5.3 repository

Content Server releases before 5.3 used the Verity full-text engine for full-text indexing. Content Server releases 5.3 and later use the index agent and index server for full-text indexing. Migrating to Content Server 5.3 or later requires that you make some decisions before migrating to the new implementation.

Note: Do not run the 5.3 SP2 or higher indexing software against a 5.2.5x or 5.3 repository except for migration purposes.

Migrating the full-text indexing system

Two paths are supported for migrating to the new full-text implementation:

- In a pre-upgrade migration, you install the new versions of the index agent and index server before upgrading Content Server. You use the new index agent and index server to create a new full-text index on the pre-5.3 repository (or a copy of the pre-5.3 repository). After the new index is created, you upgrade Content Server and the repository and use the new index. The full-text index is completely available and up-to-date after Content Server is upgraded.
- In a post-upgrade migration, you upgrade Content Server and the repository first, before installing the new index agent and index server. You create the new index by running the index agent and index server on the upgraded repository.

Pre-upgrade migration is recommended for very large repositories (100,000 documents or more) or for any repository where it is a business requirement that the full-text system is available in a consistent state immediately after the upgrade. For more information about planning for a pre-upgrade migration, refer to Migrating the full-text indexing system, page 59.

Post-upgrade migration is recommended for small repositories (fewer than 100,000 documents) or for any repository where it is acceptable for the full-text system to be in an inconsistent state immediately following the repository upgrade. The full-text system is in an inconsistent state until the new index is created. For more information on post-upgrade migration, refer to Migrating the full-text indexing system, page 59.

To perform a pre-upgrade migration of the full-text index:

1. Decide whether to index the production repository or a copy.

   To test a Content Server upgrade, it is recommended that you create a copy of the repository and test the server and repository upgrade on the copy. You can create the new full-text index by indexing either the copy or the production repository.

   If the repository is extremely large, creating new indexes takes a significant period of time. You may prefer to test the time and space required for creating new indexes on a copy or on the production repository. If you decide to index a copy of the repository, create the copy using the instructions in Chapter 4, in the section Creating a Repository Copy to Test an Upgrade.

   You are not required to create a copy of the content files, even if you create a repository copy. Instead, the file stores can be shared with the repository copy. The content can be made available to the index agent and index server in read-only mode.

   If you create the new index on the repository copy, you use the index with the production repository. When you create an index agent in normal mode, you point it to the existing index server and the production repository, and the production repository then uses the index.

2. Choose hardware for the index agent and index server.

   It is recommended that you use a host other than the Content Server host for the index agent and index server and it is strongly recommended that the hardware you choose is the hardware that will host the index and indexing software for the production repository. If you index the production repository rather than a copy, this is strongly recommended because creating new indexes is processor-intensive.

   The index agent and the index server must be installed on an operating system that is supported for 5.3 SP2. If the production repository is installed on an older operating system that is not supported for the index agent and the index server, you must install the index agent and the index server a remote host.

   The index agent and the index server may be installed on a different supported operating system from the Content Server. The index agent and index server are required to be installed on the same host.

   On Windows systems, you cannot install more than one version of DFC. The index agent requires DFC 5.3 SP2. To create the new index for the older repository, you must therefore install the index agent and index server on a host other than the Content Server host.

   On UNIX and Linux systems, you can install the index agent and index server on the Content Server host, provided the operating system is supported. You must ensure that the environment variables are set so that the existing Content Server continues to use the older DFC with which it was installed and the index agent uses DFC 5.3 SP2.

3. If you are indexing a repository copy, create the copy.

   Use the instructions in the section Creating a Repository Copy to Test an Upgrade in the *Content Server Installation Guide*.

4. Install the index server and configuration program.

   Use the instructions in Chapter 5, Installing the Full-Text Indexing Components.

5. Configure an index agent in migration mode.
6. Create the full-text indexes.

   Use the instructions in the section Creating the Full-Text Index.

7. Run the ftintegrity tool.

   Use the instructions in the section Verifying the Full-Text Indexes.

8. If any documents were not indexed, resubmit those documents for indexing.

   Use the instructions in the section Resubmitting Objects to the Index Agent.

9. Upgrade the repository.

   Use the instructions in the *Content Server Installation Guide* in the chapter Upgrading Content Server.

10. Shut down the migration-mode index agent change it to a normal-mode index agent.

    Use the Index Agent Admin Tool to switch the index agent to normal mode.

Use the following procedure to perform post-upgrade migrations of the full-text index.

To perform a post-upgrade migration of the full-text index:

1. Choose hardware for the index agent and index server.

   It is recommended that you use a host other than the Content Server host for the index agent and index server and it is strongly recommended that the hardware you choose is the hardware that will host the index and indexing software for the production repository. If you index the production repository rather than a copy, this is strongly recommended because creating new indexes is processor-intensive.

   The index agent and the index server must be installed on an operating system that is supported for 5.3 SP2. If the production repository is installed on an older operating system that is not supported for the index agent and the index server, you must install the index agent and the index server a remote host.

   The index agent and the index server may be installed on a different supported operating system from the Content Server. The index agent and index server are required to be installed on the same host.

   On Windows systems, you cannot install more than one version of DFC. The index agent requires DFC 5.3 SP2. To create the new index for the older repository, you must therefore install the index agent and index server on a host other than the Content Server host.

   On UNIX and Linux systems, you can install the index agent and index server on the Content Server host, provided the operating system is supported. You must ensure that the environment variables are set so that the existing Content Server continues to use the older DFC with which it was installed and the index agent uses DFC 5.3 SP2.

2. Upgrade the repository.
   Use the instructions in the *Content Server Installation Guide*.
3. Install the index server and index agent configuration program.
   Use the instructions in Chapter 5, Installing the Full-Text Indexing Components.
4. Configure an index agent in migration mode.
   Use the instructions in Chapter 5, Installing the Full-Text Indexing Components.
5. Create the full-text index.
   Use the instructions in Chapter 5 in the section Creating the Full-Text Index.
6. Run the ftintegrity tool.
   Use the instructions in Chapter 5 in the section Verifying the Full-Text Indexes.
7. If any documents were not indexed, resubmit those documents for indexing.
   Use the instructions in Chapter 5 in the section Resubmitting Objects to the Index Agent.
8. Shut down the migration-mode index agent change it to a normal-mode index agent.
   Use the Index Agent Admin Tool to switch the index agent to normal mode.

Migrating Verity customizations

Existing Verity thesaurus and lex files can be migrated to the new full-text indexing implementation. Refer to Importing the synonym file, page 113 for instructions on migrating a thesaurus file. Migrating a lex file requires Documentum Professional Services.

The new implementation does not use stop words. All words are indexed in order to support searching by phrase. The new implementation does not use topic trees. Existing stop files and topic trees do not need to be migrated.

Chapter 6

Installing Full-text Indexing Components

This chapter contains instructions for installing the full-text indexing software and creating full-text indexes, whether you are upgrading from an earlier Documentum release or creating new repositories. It contains the following sections:

Use the information in Chapter 3, Planning considerations, to decide on the final the index server and index agent configuration. Ensure that you have completed the tasks outlined in Chapter 4, Preparing to Install Full-Text Indexing, before installing the software.

Installing a basic deployment

This is the high-level procedure for installing the server and full-text indexing components for a new installation:

1. Choose hardware.
2. Install Content Server and configure a repository.

   Use the instructions in the *Content Server Installation Guide*.

3. Install the index server and index agent configuration program.

   Use the instructions in Chapter 5, Installing the Full-Text Indexing Components.

4. Start the full-text indexing process.

   Use the instructions in Chapter 5 in the section Creating the Full-Text Index.

5. Configure an index agent instance in normal mode.

   Use the instructions in Chapter 5, Installing the Full-Text Indexing Components.

Installing a consolidated deployment

To install a consolidated deployment, install the index agent and configure an index agent for each repository, then index each repository as described in Chapter 7, Creating and Managing the Full-Text Index.

Installing a high-availability deployment

The following instructions are for installing full-text indexing in a high-availability configuration, whether you are installing a new indexing system or converting an existing basic configuration to high availability.

In a new configuration, you must install the indexing software on the first host, perform steps on the Content Server host, install the indexing software on the second host, and perform additional steps on the Content Server host.

To convert an existing basic configuration, start with Step 2.

High-availability indexing is supported with consolidated configurations.

1. Install the full-text indexing software on the first indexing host computer and configure an index agent.

   Note: On Windows, you must log in as the same user, in the same domain, as the user who installed the Content Server installation.

   Skip this step if you are converting an existing basic configuration to high availability. Use the instructions in Installing the index server and the index agent configuration program, page 67 and Configuring the index agent, page 69.

2. Log in to the Content Server host as the installation owner.
3. Ensure that no users are connected to the repository.
4. Copy the create_fulltext_objects.ebs script from the EMC Documentum download site to the correct location on the Content Server host:

- On Windows, the %DM_DOCUMENTUM%\product\6.0\install\admin folder
- On UNIX or Linux, the $DM_DOCUMENTUM/product/6.0/install/admin directory

This version of the create_fulltext_objects.ebs script is separate from the one packaged with the Content Server distribution.

5. Shut down the first index agent.
6. Navigate to the directory where the create_fulltext_objects.ebs script is located:
   - On Windows, the %DM_DOCUMENTUM%\product\6.0\install\admin folder
   - On UNIX or Linux, the $DM_DOCUMENTUM/product/6.0/install/admin directory
7. If you are upgrading the system, run the following command:

   ```
   dmbasic -fcreate_fulltext_objects.ebs -eHACleanupBeforeUpgradeStep
   -- repository_name Superuser_name Superuser_password
   ```

   where *repository_name* is the name of the repository, *Superuser_name* is the user name of a user with Superuser privileges in the repository, and *Superuser_password* is the Superuser's password
8. Run the create_fulltext_objects.ebs script using this syntax, where *repository_name* is the name of the repository, *Superuser_name* is the user name of a user with Superuser privileges in the repository, and *Superuser_password* is the Superuser's password:

   ```
   dmbasic -fcreate_fulltext_objects.ebs -e HAPreInstallStep --
   repository_name Superuser_name Superuser_password
   ```

9. Install the full-text indexing software on the second indexing host computer and configure and index agent, using the instructions in Installing the index server and the index agent configuration program, page 67 and Configuring the index agent, page 69.

   Do not start the new index agent. The repository now contains two fulltext index objects, two ft index agent config objects, and two ft engine config objects.
10. Log in to the Content Server host as the Content Server installation owner.
11. Navigate to the directory where the create_fulltext_objects.ebs script is located:
    - On Windows, the %DM_DOCUMENTUM%\product\6.0\install\admin folder
    - On UNIX or Linux, the $DM_DOCUMENTUM/product/6.0/install/admin directory
12. Run the create_fulltext_objects.ebs script using this syntax, where *repository_name* is the name of the repository, *Superuser_name* is the user name of a user with Superuser privileges in the repository, and *Superuser_password* is the Superuser's password:

    ```
    dmbasic -f create_fulltext_objects.ebs -e HAPostInstallStep --
    repository_name Superuser_name Superuser_password
    ```

13. Restart the Content Server.
14. Using Documentum Administrator, confirm that the required objects have been created and updated.
    a. Connect to the repository as a user with Superuser privileges.
    b. Click **Indexing Management**.
    c. Click **Index Agents and Index Servers**.
    d. Verify that there are two dm_ft_engine_config objects in the repository, each representing one of the index servers.
    e. Click the Info icon for the ft engine config object representing the second index server.
    f. Write down the object ID of the ft engine config object.
    g. Click **Cancel**.
    h. In the **Starts with** box, type `dm_fulltext` and click **Go**.
    i. Verify that there are two fulltext index objects in the repository.
    j. Verify that the value of the ft_engine_id attribute in the second fulltext index object is the object ID of the ft engine config object for the second index server.
    k. Write down the object name of the second fulltext index object.
    l. Click **Cancel**.
    m. In the **Starts with** box, type `dm_ftindex` and click **Go**.
    n. Verify that an ft index agent config object exists for the second index agent.
    o. Verify that the value of the index_name attribute in the second ft index agent config object is object name of the fulltext index object for the second index and that the value of the queue_user attribute is the name of the second queue user (dm_fulltext_index_user_01).
15. Start the index agents on both hosts.
16. Create and verify the indexes on both hosts, using the instructions in Chapter 7, Creating and Managing the Full-Text Index.

Installing a multinode deployment

You must use EMC Documentum Professional Services to install and configure a multinode deployment. There are no instructions available for use without consulting Professional Services.

Installing the index server and the index agent configuration program

Use these instructions to install the index agent or the index server software. The same installation program is used for both components. You can install either or both of the components on a particular host.

Note that the installer installs the index agent configuration program, which you use to configure an index agent instance. If you do not configure the index agent immediately after installing the configuration program, you can do so at a later time.

To install the index server and the index agent configuration program:

1. Ensure that the repository for which you are installing the index server and index agent is running.
2. Log in to the index server and index agent host as the Content Server installation owner.

   Note: On Windows, this means you must log in as the same user, in the same domain, as the user who installed the Content Server installation.
3. Copy the installation files from the Documentum download site or distribution CDs to a temporary location on the host.

   The files are:

   - df*c**operatingsystem*Setup.jar
   - fulltext*operatingsystem*SuiteSetup.jar
   - fulltextWinSuiteSetup.exe (Windows) or fulltext*operatingsystem*SuiteSetup.bin (UNIX and Linux)
   - indexAgent*operatingsystem*Setup.jar
   - IndexServer.jar
   - indexServerSetup.jar
   - tomcat*operatingsystem*4127Setup.jar
4. Start the installation program.
   - On Windows, double-click fulltextWinSuiteSetup.exe.
   - On UNIX and Linux, type

     ```
     % fulltextoperatingsystemSuiteSetup.bin
     ```

     and press Enter, where *operatingsystem* is the operating system on which you are installing.

   A Welcome dialog box is displayed.
5. Click **Next**.

   The license agreement dialog box is displayed.

6. Click **I accept the terms of the license agreement** and click **Next**.

   A dialog box is displayed that lists the products you can install.

7. Choose the products to install.
   - To install the index agent, check **Documentum Index Agent Configuration Program**.
   - To install the index server, check **Index Server**.
8. Click **Next**.
9. Indicate whether to install the developer documentation and, on Windows, the primary interop assembly installer, and click **Next**.
10. If required, install Documentum Foundation Classes (DFC).
    a. On Windows, accept the default installation directory or type in a different directory and click **Next**.

       This is typically C:\Program Files\Documentum. On UNIX and Linux, the DFC directories are determined by environment variables set before installation.

    b. On Windows, accept the default user directory or type in a different directory and click **Next**.

       This is typically C:\Documentum.

11. If a dmcl.ini file does not exist on the machine, provide connection information.
    a. Type in the host name of the computer where a connection broker is running.
    b. Type in the port used by the connection broker.
    c. Click **Next**.
12. To install the index server, complete these steps.
    a. Accept the default index server installation directory or type in a new directory, then click **Next**.
    b. On Windows, type in the password for the account you used to log in, then click **Next**.

       The installer verifies the password.

    c. Type in the base port number for the index server, then click **Next**.

       The index server requires 4,000 available ports in sequence; for example, if the base port you designate is 3000, the index server uses ports 3000 through 7000. Do not use a port in ephemeral range.

       The default base port is 13000.

    d. Check the checkbox to enable support for grammatical normalization and parts of speech to be index.

Choosing the parts of speech to be indexed can reduce the size of the indexes and the disk space required for maintaining the indexes. Grammatical normalization can be enabled only for the languages listed on the dialog box. If you enable grammatical normalization, it is enabled by default for Japanese and Korean and cannot be disabled. Content files in languages that are not chosen or that are not available for normalization are still indexed.

e. Choose languages for grammatical normalization and the parts of speech to be indexed.

f. Accept the default directory for the full-text indexes or type in a different directory, then click **Next**.

The default on Windows is %DOCUMENTUM%. The default on UNIX or Linux is $DOCUMENTUM. If you choose another directory, the name must not contain any blank spaces. The installer creates the directory \data\fulltext (/data/fulltext on UNIX or Linux) under the location you designate.

A summary dialog box is displayed, listing the products that will be installed.

13. Click **Next**.

The products are installed and a panel is displayed indicating success when the installation is completed.

14. Ensure that the index server starts.

- On Windows, select **Yes, restart my computer.**, then click **Next**.
  - If the computer does not restart automatically, click **Start→Shutdown→Restart** and restart the computer manually.
  - If the index server does not automatically start, click **Start→Programs→Administrative Tools→Services** and start the FAST InStream service.
  - If the system restarts, the index server starts automatically as a Windows service.
- On UNIX and Linux, navigate to the $DOCUMENTUM/fulltext/IndexServer/bin directory (the installation location), type `startup.sh`, and press Enter.

The index server is started.

Configuring the index agent

The index agent configuration program configures the index agent to process documents for a particular repository and to pass the documents to the correct index server instance for indexing. Use these instructions to run the index agent configuration program.

To configure the index agent:

1. Ensure that the index server associated with the index agent is running.
2. Start the index agent configuration program.
   - On Windows, after the host reboots and you log in as the installation owner, click **Start→Programs→Documentum→Index Agent Configuration Program.**
   - On UNIX and Linux, navigate to $DOCUMENTUM_SHARED/IndexAgents and start the configuration program for your operating system:
     - On AIX, IndexAgent_Configuration_Program.aix
     - On Solaris, IndexAgent_Configuration_Program.bin
     - On HP-UX, IndexAgent_Configuration_Program.hp
     - On HP Itanium, IndexAgent_Configuration_ProgramHPIA64.bin
     - On Linux, IndexAgent_Configuration_Program.linux

   A Welcome dialog box is displayed.
3. Click **Next**.
4. On Windows, type in the installation owner's password, then click **Next**.
5. Type in the ports that the index agent uses for communicating with Tomcat and for stopping Tomcat, then click **Next**.

   The default ports are 9081 and 9008 for the first index agent on the host. If the index agent is on the Content Server host, the port numbers must not be the port numbers used by the Java method server or Site Caching Services.
6. Select the repository for which the index agent will prepare documents, then click **Next**.

   The drop-down list contains the repositories that project to the connection brokers listed in the dmcl.ini file on the host. The dmcl.ini file was created during installation if there was not already a dmcl.ini file present on the host.
7. Type in the user name and password for the Superuser account that the index agent will use to connect to the repository.

   Use this user name and password later to access the Index Agent Admin Tool.
8. If the index agent is running against a 5.3 SP1 or later repository, indicate whether to configure it in normal mode or migration mode.
9. Type in the host where the index server for this index agent is running and the base port number for the index server, then click **Next**.

   If you are configuring the index agent for the second indexing installation in a high-availability deployment, ensuring that you point the index agent to the second index server host, not the first index server host. The installer defaults to the first index server host.

A summary dialog box is displayed.

10. Click **Next**.

    On Windows, the index agent is created and Tomcat is started.

11. If you are on UNIX or Linux, navigate to $DOCUMENTUM_SHARED/IndexAgents/IndexAgent*N*/, where *N* is the number corresponding to the new index agent instance, and type startupIndexAgent.sh to start the index agent and its Tomcat instance.

12. To create additional index agents, select the checkbox and click **Next**, then complete steps 3–9 again.

13. To exit from the configuration program, click **Finish**.

14. To complete the installation process:
    - If you are mapping the file stores, complete the instructions in Modifying the indexagent.xml file to map file stores, page 71.
    - If you are installing a high-availability configuration and you have installed the index server and configured the index agent on the first host, return to Installing a high-availability deployment, page 64 and continue from Step 2.
    - If you are installing a high-availability configuration and you have installed the index server and configured the index agent on the second host, return to Installing a high-availability deployment, page 64 and continue from Step 9.

      Do not start the index agent on the second host.

Modifying the indexagent.xml file to map file stores

If you have shared or mounted the drives containing the repository's file stores and installed the indexing software, the index agent configuration file must be manually edited to indicate that the drives are shared. The changes depend on whether the file system paths to the content are identical on the Content Server host and index server host.

In a distributed configuration, all content located in distributed stores is moved to the primary site, using the Surrogate Get method, and indexed at the primary site. You cannot map the remote components of a distributed store. Similarly, you cannot map an encrypted store or an external store.

To modify the indexagent.xml file and map the file stores:

1. On the index agent host, navigate to $DOCUMENTUM_SHARED\IndexAgents\IndexAgent1\webapps\IndexAgent1\WEB-INF\classes\.
2. Open the indexagent.xml file in a text editor.

3. If the paths to the content files are identical on the Content Server host and index server host, locate the <exporter></exporter> element and change the value of the <all_filestores_local> element to true:

   ```
   <all_filestores_local>true</all_filestores_local>
   ```

4. If the paths to the content files are different, do not modify the value of <all_filestores_local>, but instead, create a file store map within the <exporter> element.

   For example, if Content Server is on a host called Dandelion where filestore_01 is physically located in the directory /Dandelion/Documentum/data/*repository_name*/content_storage_01 and the index agent and index server on a host from which the drive on the Content Server host is shared as /mappingtoDandelion/*repository_name*/content_storage_01, create an alias as follows:

   ```
   <local_filestore_map>
     <local_filestore>
      <store_name>filestore_01</store_name>
      <local_mount>/mappingtoDandelion/repository_name
              /content_storage_01</local_mount>
     </local_filestore>
      <!-- and so on for each filestore --!>
    </local_filestore_map>
   ```

   If you are indexing content stored on a NAS device or a Windows 2003 Server host, you may see the following error message in the dmi_queue_item's message attribute:

   ```
   DocumentRetriever :ERROR Retrieval error: Couldn't open
   file <file path/name> ERROR Processor error status:
   DataNotAvailable Not read permission
   ```

   To resolve this error, edit the <local_mount> element or elements in the IndexAgent.xml file that reference the storage area or areas on the NAS device. Add two backslashes immediately after the opening <local_mount> element. For example, assume the following references a storage are on an NAS device:

   ```
   <local_mount>\\100.2.4.32\share3\c\data_for_example
          \content_storage_1</local_mount>
   ```

   After editing, it is now:

   ```
   <local_mount>\\\\100.2.4.32\share3\c\data_for_example
       \content_storage_1</local_mount>
   ```

5. Save the indexagent.xml file.
6. Restart the index agent.

Reviewing the installation log files

On any host where you installed the full-text indexing components, an installer log is generated. Navigate to the directory from which you ran the installation and examine install.log for errors and warnings.

Chapter 7

Creating and Managing the Full-Text Index

This chapter contains the information you need to create and verify the accuracy and completeness of the full-text index. The chapter contains the following topics:

If you are installing a high-availability configuration, create the and verify the index on both hosts.

Creating the full-text index

The operations of the index agent and index server are controlled through a Web interface located at http://*hostname*:*portno*/IndexAgent*N*/login.jsp, where *hostname* is the name of the host where the index agent is running, *portno* is the port where the index agent is listening, and *N* is the number designating a particular index agent instance. (The port number is the port you provided during index agent configuration.) You can stop and start the index agent or index server, or view the status of the index agent or index server from the Web page.

It is recommended that you put the index server into suspended mode while objects from the repository are being processed. Suspended mode speeds that process; when it is completed, you put the index server into indexing mode and the index is updated.

Updating an index may take a long period of time, depending on the number of objects to be added to the index, the hardware on which the index server runs, and the size of the objects submitted for indexing. The index server does not notify the index agent and Content Server when the index becomes searchable. Thus, there is no way to know the exact time at which an index can be searched or a particular document has been added to the index. If a repository is being indexed for the first time, it can take hours before the index is fully updated and searchable. If the ftintegrity tool is run while the index is being updated, the tool reports many discrepancies. The index is not searchable until the update is completed.

Submitting objects for indexing

Use these instructions to submit objects for indexing, which starts the process of creating full-text indexes for the target repository. These instructions apply whether the index agent is running in migration mode or normal mode. In normal mode, you must provide the index agent instance name because the index agent is controlled by a configuration object in the repository.

To submit objects for full-text indexing:

1. Ensure that you have performed the instructions in Modifying the indexagent.xml file to map file stores, page 71.
2. Start a browser and point it to this URL:

   `http://`*`hostname`*`:`*`portno`*`/IndexAgent`*`N`*`/login.jsp`

   where *hostname* is the name of the host where the index agent is running and *portno* is the port where the index agent is listening and *N* is the number assigned to the index agent instance. If the browser is on the index agent host, replace *hostname* with `localhost`.
3. Log in to the Index Agent Admin Tool using the credentials for the user who connects to the repository for indexing operations.
4. To put the index server into suspended mode, which improves indexing performance, click **Switch Mode**.

   It is strongly recommended that you put the index server into suspended mode.
5. Click **Ok**.
6. In the index agent status line, click **Start**.
7. Click **Ok**.

The indexing software begins the indexing process.

8. If you put the index server into suspended mode, when the objects in the repository are processed, click **Switch Mode** and put the index server into indexing mode.

Caution: If a large set of documents is submitted for indexing, it may take hours for indexing to complete and the index to be available for searching. If you run the ftintegrity tool prematurely or attempt to search the index before indexing is complete, the results are incorrect or misleading, and the index may appear corrupt when it is not yet complete.

Stopping full-text indexing

Use these instructions to stop the process of creating full-text indexes. These instructions apply whether you are running the index agent in migration mode or normal mode. If the index agent and index server are running against a 5.3 SP2 repository, you can also shut them down using Documentum Administrator.

To stop submitting objects for full-text indexing:

1. Start a browser and point it to this URL:

   `http://`*`hostname`*`:`*`portno`*`/IndexAgent`*`N`*`/login.jsp`

   where *hostname* is the name of the host where the index agent is running and *portno* is the port where the index agent is listening and *N* is the number assigned to the index agent instance. If the browser is on the index agent host, replace *hostname* with `localhost`.
2. Log in to the Index Agent Admin Tool.
3. In the index agent status line, click **Stop**.
4. Click **Ok**.
5. In the index server status line, click **Stop**.
6. Click **Ok**.
7. Click **Go**.

Note that stopping the index agent and index server does not shut down the containers in which the applications run. The index agent's Tomcat instance and the index server's Apache Web server instance are still running.

Checking the status of the index agent

Use these instructions to display status information about the index agent, including the name of the repository with which the index agent is associated, the mode of operation (migration or normal), the number of documents in each batch that the index agent processes (agent_batch_size), the interval at which the index agent polls the repository for new documents to be indexed, and the current status.

To check the status of the index agent:

1. Start a browser and point it to this URL:

   `http://`*`hostname`*`:`*`portno`*`/IndexAgent`*`N`*`/login.jsp`

   where *hostname* is the name of the host where the index agent is running and *portno* is the port where the index agent is listening and *N* is the number assigned to the index agent instance. If the browser is on the index agent host, replace *hostname* with `localhost`.
2. Log in to the Index Agent Admin Tool.
3. Click **Configure**.

   Information about the index agent is displayed.

Managing the index queue

In 5.3 and later repositories, creating, versioning, or deleting a SysObject or SysObject subtype creates a queue item indicating that the full-text indexes must be updated to account for the changes. The index agent reads items from the queue and ensures that the required index updates take place.

If the repository's indexing system runs in a high-availability configuration, with multiple index agents and index servers, each index agent/index server pair supports its own index. Creating, versioning, or deleting a SysObject or SysObject subtype creates a queue item for each pair and each index is updated.

Documentum Administrator provides the interface for viewing the current index queue and managing the items in the index queue.

The **Indexing Management→Index Queue** page lists index queue items for the current repository. To sort the queue items, click the **Object ID**, **Object Name**, **Object Type**, **Task Status**, **Acquired by**, or **Creation Date** links. The **Object Name** and **Object ID** columns list the object name and object ID of the object to be indexed, not the index queue item. If you click the Info icon for a queue item, the properties of the object to be indexed are displayed, not the queue item's properties.

If the indexing system is in a high-availability configuration, the name of each index is displayed at the top of this page and only the queue items for one index at a time are displayed. To change which index's queue items are displayed, click the name of index.

To change the number of queue items displayed, select a different number in the **Show Items** drop-down list.

By default, the list page displays failed queue items. To filter the queue items by status, choose the appropriate status on the drop-down list:

- **Indexing Failed**, which is the default status displayed

  If indexing failed, information about the error is displayed in red under the queue item's name and other properties.
- **All**, which displays all current queue items in the repository
- **Indexing in Progress**, which indicates that the object is being processed by the index agent or index server
- **Awaiting Indexing**, which indicates that the index agent has not yet acquired the queue item and started the indexing process
- **Warning**, which indicates that the index agent encountered a problem when it attempted to start the indexing process for the object

  If indexing generated a warning, information about the problem is displayed in red under the queue item's name and other properties.

Queue items that have failed indexing can be resubmitted individually, or all failed queue items can be resubmitted with one command.

The procedures in this section cover the following topics:

- Resubmitting individual objects, page 79
- Resubmitting all failed queue items, page 80
- Removing queue items by status, page 80
- Removing queue items, page 81
- Viewing queue items associated with an object, page 81
- Creating a new indexing queue item, page 81

Resubmitting individual objects

You can resubmit individual objects for indexing.

To resubmit individual objects:

1. Connect to the repository as user who has System Administrator or Superuser privileges.

2. Click **Administration>Indexing Management>Index Queue**.
3. If the repository's indexing system is installed in a high-availability configuration, ensure that the index queue for the correct index is selected.
4. Choose an object.
5. Click **Tools>Resubmit Queue Item**.

Resubmitting all failed queue items

You can resubmit for indexing all documents that failed indexing. This menu choice executes the mark_for_retry administration method. If the indexing system is installed in a high-availability configuration, all failed queue items for all indexes are resubmitted.

To resubmit all objects that failed indexing:

1. Connect to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Queue**.
3. Click **Tools>Resubmit all failed queue items**.

Removing queue items by status

Use these instructions to remove index queue items by status.

To remove queue items by status:

1. Connect to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Queue**.
3. If the repository's indexing system is installed in a high-availability configuration, ensure that the index queue for the correct index is selected.
4. Click **Tools>Remove Queue Items by Status**.
5. Select the correct status.

Removing queue items

Use these instructions to remove queue items from the indexing queue. Note that if a queue item has already been acquired by the index agent, it cannot be removed from the indexing queue.

To remove queue items:

1. Connect to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Queue**.
3. If the repository's indexing system is installed in a high-availability configuration, ensure that the index queue for the correct index is selected.
4. Select the queue items.
5. Click **Tools>Remove queue items**.

Viewing queue items associated with an object

From a repository's cabinets, you can view the index queue items associated with a particular object.

To view the queue items associated with an object:

1. Connect to the repository as user who has System Administrator or Superuser privileges.
2. Navigate to the object in the repository's cabinets.
3. Select the object.
4. Click **Tools>View Queue Items**.

   The queue items are displayed for the selected index queue.
5. If the repository's indexing system is installed in a high-availability configuration, optionally click the links for each index queue.

Creating a new indexing queue item

You can create a queue item to submit a particular SysObject for indexing.

To create a queue item and submit and a particular object for indexing:

1. Connect to the repository as user who has System Administrator or Superuser privileges.
2. Navigate to the object in the repository's cabinets.
3. Select the object.
4. Click **File>New>Create queue item**.
   The index queue and new queue item are displayed.
5. If the repository's indexing system is installed in a high-availability configuration, optionally click the links for each index queue.

Limitations of full-text indexing in high-availability configurations

The following sections describe current limitations in high-availability full-text indexing configurations.

The Prune API and missing Destroy events

The Prune API is used to delete multiple object versions from a version tree. The Prune API generates a Destroy event for each version that is deleted. Content Server can only generate these Destroy events for the default full-text indexing queue users. The events cannot be generated for the second indexing queue user in a high-availability configuration.

This means that when the Prune API is used, the index entries for the objects destroyed cannot be removed from the nondefault index. However, false positive hits on the index for such objects are filtered from query results, and are not returned to end users or applications generating the queries.

Save events not generated during load operations

Load operations do not generate save events for objects created during the load; special events are generated instead. The events cannot be queued to the second full-text user.

Verifying index completeness and accuracy

Use the ftintegrity utility to verify the completeness or accuracy or both of an index. Using this tool after completing an index migration or adding an index to an existing repository is recommended.

When run in completeness mode, the utility generates three reports:

- res-comp-common.txt

  This file contains the object IDs of all documents that are found in both the index and in the repository.

- res-comp-dctmonly.txt

  This file contains the object IDs of all indexable documents that are found in the repository but not in the index.

  Note: The ftintegrity tool is not aware of object types that you have chosen not to index, through either a custom filter or the specification of selective indexing in Documentum Administrator. If you have chosen not to index one or more object types, the object IDs of instances of those types will appear in this file.

- res-comp-fastonly.txt

  This file contains the object IDs of documents found in the index, but not in the repository.

When running in accuracy mode, you can check a subset of documents or all documents in the index. For each document the utility performs the following operations:

1. Gets metadata from the repository
2. Chooses a random metadata item and generates a random test for that item.

   For example, it might generate the query: subject like “%foo”, r_content_szie>4000
3. Runs a test program named “iftdql’ to find that document.

The utility is installed when the index agent is configured. There is one instance of the utility for each index agent. Each instance has an associated parameter file, called ftintegrity.params.txt. This file identifies the repository on which the utility operates. It also parameters, which you must complete before running the utility, that identify the user name and password used to connect to that repository. The utility and its associated parameter file are found in:

- On Windows

  Drive:\Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent*N*

- On UNIX and Linux

  indexagentinstallationdir/IndexAgents/IndexAgent*N*/webapps/IndexAgent*N*

Note: Each index agent you configure after the first is provided with a numbered directory. For example, the second index agent found in \Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent2. You can determine which repository each index agent is associated with by examining the indexagent.xml file in those subdirectories or the agentinstances.xml file the IndexAgents directory. That file records the repository name, associated index agent, and port numbers.

The utility is run from the command line. Before running the utility, you must modify its associated parameter file to add the user name and password. For instructions on modifying the file, refer to Modifying the parameter file, page 84. For instructions on running the utility, refer to Running the index verification tool, page 85.

After you run the utility, you may wish to resubmit documents for indexing if the utility reports missing documents. For resubmission instructions, refer to Resubmitting objects to the index agent, page 87.

Modifying the parameter file

Before running the index verification tool, modify the parameter file to include login information for the repository for which you created the indexes.

Caution: The instruction below save a Superuser name and password to the file system in a plain text parameter file. For security reasons, you may wish to remove that information from the file after running the ftintegrity tool. It is recommended that you save the parameter file in a location accessible only to the repository Superuser and installation owner.

To modify the parameter file:

1. Navigate to the parameter file location.
   - On Windows,*Drive*:\Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent*N*
   - On UNIX and Linux, *indexagentinstallationdir*/IndexAgents/IndexAgent*N*/webapps/IndexAgent*N*
2. Open the ftintegrity.params.txt file in a text editor.

   The first line is

   ```
   -D repositoryname
   ```

   where *repositoryname* is the repository for which you created a new index.
3. Add the following two lines immediately after the first line

   ```
   -U username
   -P password
   ```

where *username* is the user name of the Superuser whose account was used to install the index agent and *password* is the Superuser's password.

4. Save the ftintegrity.params.txt file to %Documentum%\fulltext\IndexServer\bin (Windows) or $Documentum/fulltext/IndexServer/bin (UNIX)..

Running the index verification tool

Use these instructions for running the index verification tool. If you are on UNIX or Linux, the tool can be started by running a script that sets the required environment variables. The script is called run_ftintegrity.sh, and it is located in $DOCUMENTUM/fulltext/IndexServer/bin. You must pass the same arguments regardless of whether you use the script or not.

To run the index verification tool:

1. Navigate to %Documentum%\fulltext\IndexServer\bin (Windows) or $Documentum/fulltext/IndexServer/bin (UNIX).
2. To verify both completeness and accuracy, open a command prompt and type

   ```
   cobra ftintegrity.py -i ftintegrity.params.txt -m b
   ```

   then press Enter.

   Note: The ftintegrity tool is not aware of object types that you have chosen not to index, through either a custom filter or the specification of selective indexing in Documentum Administrator. If you have chosen not to index one or more object types, the object IDs of instances of those types will appear in the generated res-comp-dctmonly.txt file.
3. To verify completeness only, open a command prompt and type

   ```
   cobra ftintegrity.py -i ftintegrity.params.txt -m c
   ```

   then press Enter.

   Note: The ftintegrity tool is not aware of object types that you have chosen not to index, through either a custom filter or the specification of selective indexing in Documentum Administrator. If you have chosen not to index one or more object types, the object IDs of instances of those types will appear in the generated res-comp-dctmonly.txt file.
4. To verify accuracy only and query all indexed objects, open a command prompt and type

   ```
   cobra ftintegrity.py -i ftintegrity.params.txt -m a
   ```

   then press Enter.

5. To verify accuracy only and query a subset of indexed objects, open a command prompt and type

   ```
   cobra ftintegrity.py -i ftintegrity.params.txt -m a -d integer
   ```

   where *integer* is the number of objects in the subset, then press Enter.

   If the integrity tool runs successfully, the screen output is similar to this sample:

```
E:\Documentum\fulltext\IndexServer\bin>cobra ftintegrity.py -i
      ftintegrity.params.txt -m b
Wed Mar 23 13:33:38 2005 [PROGRESS]: Completeness Test Started
Wed Mar 23 13:33:47 2005 [PROGRESS]: Fetched 10000 docids from Documentum
Wed Mar 23 13:33:53 2005 [PROGRESS]: Fetched 20000 docids from Documentum
Wed Mar 23 13:33:58 2005 [PROGRESS]: Fetched 30000 docids from Documentum
Wed Mar 23 13:34:00 2005 [PROGRESS]: Fetched all 31261 docids from Documentum
Wed Mar 23 13:34:00 2005 [INFO    ]: Get Documentum IDs: 21.870
Wed Mar 23 13:34:00 2005 [DEBUG   ]: Got 31261 docids from Dctm
Wed Mar 23 13:34:00 2005 [INFO    ]: Get Number of Documents in
      Collection: 0.241 s
Wed Mar 23 13:34:00 2005 [INFO    ]: Number of documents in collection
      documentum: 502
Wed Mar 23 13:34:01 2005 [PROGRESS]: Fetched all 502 docids from index server
Wed Mar 23 13:34:01 2005 [INFO    ]: Get index server IDs: 0.771 s
Wed Mar 23 13:34:01 2005 [DEBUG   ]: Got 502 docids from index server
Wed Mar 23 13:34:01 2005 [INFO    ]: Transform index server ID file: 0.005 s
Wed Mar 23 13:34:01 2005 [INFO    ]: Transform Documentum ID file: 0.270 s
Wed Mar 23 13:34:01 2005 [INFO    ]: Compare IDs: 0.271 s
Wed Mar 23 13:34:01 2005 [PROGRESS]: Completeness Test Finished
Wed Mar 23 13:34:01 2005 [PROGRESS]: Accuracy Test Started
Wed Mar 23 13:34:01 2005 [DEBUG   ]: Getting Dctm Ids from res-comp-common.txt
Wed Mar 23 13:34:01 2005 [PROGRESS]: Fetched all 502 docids from Documentum
Wed Mar 23 13:34:01 2005 [PROGRESS]: Starting test of 0 documents
Wed Mar 23 13:34:01 2005 [PROGRESS]: Launching iftdql: C:\PROGRA~1\DOCUME~1
  \INDEXA~1\FTINTE~1\iftdql C:\PROGRA~1\DOCUME~1\INDEXA~1\FTINTE~1
Wed Mar 23 13:34:10 2005 [PROGRESS]: Tested 100 of 502 documents,
   0 failures. Estimated completion in 0.62 minutes
Wed Mar 23 13:34:13 2005 [PROGRESS]: Tested 200 of 502 documents,
   0 failures. Estimated completion in 0.31 minutes
Wed Mar 23 13:34:17 2005 [PROGRESS]: Tested 300 of 502 documents,
   0 failures. Estimated completion in 0.18 minutes
Wed Mar 23 13:34:20 2005 [PROGRESS]: Tested 400 of 502 documents,
   0 failures. Estimated completion in 0.08 minutes
Wed Mar 23 13:34:23 2005 [PROGRESS]: Tested 500 of 502 documents,
   0 failures. Estimated completion in 0.00 minutes
Wed Mar 23 13:34:24 2005 [INFO    ]: Total documents:  502
Wed Mar 23 13:34:24 2005 [INFO    ]: Documents tested: 502
Wed Mar 23 13:34:24 2005 [INFO    ]: Failures:         0
Wed Mar 23 13:34:24 2005 [INFO    ]: Statistical Information:
Wed Mar 23 13:34:24 2005 [INFO    ]:        Min        Max
    Avg     StdDev  N Name
Wed Mar 23 13:34:24 2005 [INFO    ]:       0.02       5.01
    0.03       0.22 502 Dump Attributes
Wed Mar 23 13:34:24 2005 [INFO    ]:       0.00       0.00
    0.00       0.00 502 Build Query
Wed Mar 23 13:34:24 2005 [INFO    ]:       0.01       0.37
    0.01       0.03 502 Execute Query
Wed Mar 23 13:34:24 2005 [INFO    ]:       0.02       5.38
```

```
    0.04       0.24 502 Test One Document
Wed Mar 23 13:34:24 2005 [PROGRESS]: Accuracy Test Finished
E:\Documentum\fulltext\IndexServer\bin>
```

Additional samples of the tool's output are in Appendix C, Sample Output of ftintegrity Utility.

Accuracy testing confidence and failures

Running the accuracy testing provides a high degree of confidence that the index is accurate, but the testing is not exhaustive. Consequently, there may be some acceptable failures. For example, the following causes of failures may be acceptable:

- Missing r_object_id values.

  This may occur if an object ID was not indexed. However, ftintegrity tool is not aware of object types that you have chosen not to index, through either a custom filter or the specification of selective indexing in Documentum Administrator. If you have chosen not to index one or more object types, the object IDs of instances of those types will appear in the generated res-comp-dctmonly.txt file.
- vstamp mismatches

  The object in the index may not yet reflect any changes made in the repository.
- Transient failures due to intermittent resource problems experienced by the index server
- Failure to dump property values, possibly due to inactive Content Server
- Metadata query failures related indexing of special characters

Resubmitting objects to the index agent

When the ftintegrity tool is run, the completeness check produces a text file of object IDs corresponding to objects that are found in the repository but not in the full-text index. The file is called res-comp-dctmonly.txt. It is located in the Documentum/fulltext/indexserver/bin directory. Use the file to resubmit objects to the index agent for indexing.

Because the ftintegrity tool is not aware of object types that you may have chosen not to index, through either a custom filter or the specification of selective indexing in Documentum Administrator, the object IDs of instances of those types will appear in the generated res-comp-dctmonly.txt file. However, when you use the file to resubmit objects for indexing, the objects represented by those object IDs are ignored. If you are using a custom filter to define the nonindexed types, the operation will generate queue

items for those objects, but they are not indexed and the queue items are deleted later in the process.

Objects may be resubmitted when the index agent is running in either migration mode or normal mode. The file of object IDs is designated in the file_name parameter of the indexagent.xml configuration file for a particular index agent. By default, the file_name parameter is set to the file name ids.txt. The indexagent.xml file is in *drive*:\Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent*N*\WEB-INF\classes (Windows) or *indexagentinstalldirectory*/IndexAgents/IndexAgent*N*/webapps/IndexAgent*N*/WEB-INF/classes (UNIX), where *N* is the number of the particular index agent.

The index agent periodically checks for the existence of ids.txt. If the file is found, the objects are resubmitted for indexing.

The file designated in the file_name parameter can have any arbitrary name. For example, the output file re-comp-dctmonly.txt might be renamed to resubmit.txt or to the default ids.txt. The name may be a simple name (for example, ids.txt) or a fully-qualified name (for example, C:\Program Files\Documentum\MyFiles\ids.txt). If the file has a simple name, the index agent tries to resolve the name by following the CLASSPATH. If you change the name from the default name, you must modify the indexagent.xml file and substitute the name you assign the file in the file_name parameter.

A particular file must be made available to one index agent only. If multiple index agents are installed on a host, at installation time the indexagent.xml files for all index agents name the ids.txt file in the file_name parameter.

For example, if the file is located in C:\Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent*N*\WEB-INF\classes, only that instance will find the file.

When the index agent has read the entire file and submitted all objects for indexing, the input file is renamed to *original_filename*.done. For example, if the input file is ids.txt, when all objects are submitted, it is renamed ids.txt.done.

To resubmit objects to the index agent:

1. Run the ftintegrity tool.
2. Navigate to $DOCUMENTUM//fulltext/indexserver/bin (%DOCUMENTUM\fulltext\indexerver\bin).
3. Copy the res-comp-dctmonly.txt file to *indexagentinstalldirectory*/IndexAgents/IndexAgent*N*/webapps/IndexAgent*N*/WEB-INF/classes (*drive*:\Program Files\Documentum\IndexAgents\IndexAgent*N*\webapps\IndexAgent*N*\WEB-INF\classes).
4. Rename the res-comp-dctmonly.txt file to ids.txt or to an arbitrary name you choose.

5. If you rename res-comp-dctmonly.txt to any name other than ids.txt, you must modify the file_name parameter in the indexagent.xml file for the particular index agent to reflect the new name.

   The objects are automatically resubmitted for indexing.

The State of the Index job

Configuring a repository installs a job called State of the Index. This job is implemented as a Java method. Running the job generates reports that provide the following information:

- Index completeness information, similar to ftintegrity, but also comparing version stamps on documents
- Status information about the index server, including disk space usage, node statistics, and process statuses
- Information about total number of objects with content correctly indexed, total number of objects with content that had some failure during indexing, and total number of objects with no content

If you have configured a custom filter to ignore specific object types when indexing, you can include the -usefilter argument to ensure that the job also uses that filter to exclude objects of those types from the generated reports.

The job does not recognize object types chosen for exclusion from indexing through Documentum Administrator. Objects of those types will be included in the generated report. If you have a custom filter configured to exclude object types from indexing, you can use the -usefilter argument to exclude instances of those types from the reports generated by the job.

Execute the job from Documentum Administrator.

Arguments

Table 7–1, page 90, lists the arguments for the job.

Table 7-1. State of the Index job arguments

Argument	Datatype	Description
-batchsize *value*	integer	Number of objects to be retrieved from the index in each batch. The default value is 1000.
-writetodb_ threshold *value*	integer	Determines how the object comparison is conducted. If the total number of returned objects is less than the value of this argument, or its default value, the comparison is conducted in-memory. If the total number is greater, the comparison is conducted in the database. The default value is 100000. **Note:** Conducting the comparison in memory consumes memory and in a limited system, may result in an out-of-memory error.
-usefilter *value*	Boolean	Determines whether a filter is used during the object comparisons. The default value is F. If a custom filter is used for indexing, set this argument to T to use the filter when generating the job results. If this is set to T but no filter is found, the job ignores this argument, but logs a message in the job report. **Note:** Custom filters can be created to stop indexing of specific object types. Information is available on the developer.com website.
-dumpfailedid *value*	Boolean	Indicates whether to generate the docids-failedids.txt file, which records the object IDs of objects with content that experienced some failure during content indexing. T means to generate the file. F means do not generate that file. The default is F.
-serverbase *value*	Boolean	Whether to collect information about objects with content and objects without content from one collection in the index or all collections. T means to collect information from all collections. F means to collect information from only one collection.

Argument	Datatype	Description
		The default is F.

In addition, the job is installed with the -queueperson, -windowinterval, and -max_events_per_run arguments set. The -queueperson and -windowinterval arguments are standard arguments for administration jobs, and are explained in the *Content Server Administrator's Guide*. The -max_events_per_run argument is not currently used.

Job report and generated files

The job generates a job report, FTStateofIndexDoc.txt and four results files. The FTStateofIndexDoc.txt contains information about the job execution, similar to the job reports generated by other administration jobs. The four results files are:

- docids-common.txt

  This file contains the object IDs and i_vstamp values of all objects that are found in both the index and the repository and having identical i_vstamp values in both places.
- docid-common-misVstamp.txt

  This file records all objects that are found in both the index and the repository with identical object IDs but non-matching i_vstamp values. For each object, it records the object's object ID, i_vstamp value in the repository, and i_vstamp value in the index.
- docids-dctm-only.txt

  This report contains the object IDs and i_vstamp values of objects that found in the repository but not in the index.
- docids-fast-only.txt

  This report contains the object IDs and i_vstamp values of objects that found in the index but not in the repository.

In addition to the report and the four standard result files, the report may generate additional result files depending on the arguments defined for the job run:

- If -dumpfailedid is set to T, the job generates a file called docids-failedids.txt. This file contains the object ID of all objects whose content experienced some failure duriing indexing.
- If -usefilter is set to T, the job generates two additional reports.
  - docids-filtered.txt, which records the objects that have been filtered from the repository for indexing. The objects are identified by object ID and i_vstamp values.

— docids_shouldnotin_fast.txt, which records the object ID and i_vstamp values of any objects in the index that should have been filtered out before indexing.

The report and result files are found in %DOCUMENTUM%\dba\log*sessionID*\sysadmin ($DOCUMENTUM/dba/log/*sessionID*/sysadmin).

Creating indexing events for new content in a repository

In some circumstances, you must ensure that events are generated for content added to a repository between the time a full-text index is created and the time the repository is upgraded.

For example, a copy of a 5.2.5 repository can be used to create a new 5.3 SP2 index. Depending on when the production repository is upgraded, new indexable objects may be created in the 5.2.5 production repository after the new 5.3 SP2 index is created on the repository copy. When the production repository is upgraded to 5.3 SP2 and begins to use the new index, the repository contains objects that are not yet indexed. A job installed by default in all 5.3 and later repositories, dm_FTCreateEvents, can be used in its default mode to generate events for new indexable objects. An index agent running in normal mode uses the events to submit the objects for indexing.

By default, the job is installed in the active state to run daily at 11 p.m. The job processes new objects in batches of 50,000. If all objects are not processed in one run, the job continues executing each day at 11 p.m. When it finds no more objects to process, it sets itself to inactive.

The first time the job runs, the job determines the last object indexed by an index agent running in migration mode and the date on which that object was indexed. The job searches for objects modified after that date and before the job runs for the first time and generates events for those objects. On its subsequent iterations, the job searches for objects modified after the end of the last iteration and before the beginning of the current iteration.

After a repository upgrade, use Documentum Administrator to review the job log for dm_FTCreateEvents. The dm_FTCreateEvents job can also be run manually using Documentum Administrator.

Turning indexing on and off

The following guidelines apply to turning indexing on and off:

- You can turn off indexing of both properties and content files.

  Turning off all indexing, page 93, contains instructions for turning off indexing.
- You can turn off content indexing but leave properties indexed.

  Turning off content indexing, page 93, contains instructions for turning off content indexing.
- You cannot turn off only property indexing.
- You can disable the creation of the full-text index by shutting down the index agent and index server for a repository, but queue items are still generated by the events that trigger indexing.

Turning off all indexing

To turn off all full-text indexing operations in a repository, connect as a Superuser, then unregister the Save, Checkin, Destroy, Readonlysave, and MoveContent notifications for the full-text user (dm_fulltext_index_user). This disables the generation of queue items for Save, Checkin, Destroy, Readonlysave, and MoveContent events.

Turning off content indexing

To turn off content indexing, set the a_full_text property of the SysObject to FALSE.

Setting a_full_text to FALSE turns off content indexing but does not turn off indexing of the object's properties.

Suspending and resuming indexing

Suspending indexing stops the index server from processing the FIXML files generated by the indexing process. The generation of FIXML files in not stopped. The process proceeds through the generation of the FIXML files, but is suspended at that point. When indexing is resumed, the index server resumes processing the FIXML files and updating the index.

An index server must be running when you suspend its indexing functions.

Suspending and resuming an index server in a single-node configuration

Use the following command to suspend an index server:

```
rtsadmin adminhost port webcluster 0 0 suspendindexing
```

To resume indexing, use the following command:

```
rtsadmin adminhost port webcluster 0 0 resetindex
```

where *adminhost* is the name of the index server's administrative host. The *port* is the index server's base port number plus the constant 3099. If the index server is using the default base port, then the *port* value is 16099.

Suspending and resuming an index server in a multinode configuration

In a multinode configuration, you issue one suspend or resume command for each node. For example, if there are two nodes in the configuration, you use the following commands:

```
rtsadmin adminlhost port webcluster 0 0 suspendindexing
rtsadmin adminhost port webcluster 1 0 suspendindexing
```

To resume indexing on those nodes, use the following commands:

```
rtsadmin adminhost port webcluster 0 0 resetindex
rtsadmin adminhost port webcluster 1 0 resetindex
```

where *adminhost* is the name of the index server's administrative host. The *port* is the index server's base port number plus the constant 3099. If the index server is using the default base port, then the *port* value is 16099.

The first integer after 'webcluster' represents a column, and each node has one column. The commands to suspend and reset indexing reference the nodes through the column numbers. Column numbers start with 0 and increment by one for each columns, or node. The second integer must always be 0.

To determine how many nodes are running, if needed, you can use the dsadmin tool to find all the configured nodes. You can use grep (UNIX) or findstr (Windows) to limit the returned values to Indexer modules. For example:

```
$ dsadmin listmodules | grep Indexer
[2006-12-11T21:47:17Z]: INFO  dsadmin RTS Indexer    3.0.98-Release
       mice.mylabs.com    15674
[2006-12-11T21:47:17Z]: INFO  dsadmin RTS Indexer    3.0.98-Release
       cats.mylabs.com    15674
```

Count the number of returned rows to determine how many indexing nodes are running. This example return indicates that there are two nodes.

Configuring the indexing behavior

This section describes how to configure indexing behaviors, to help ensure that users' searches find the objects they want.

Disabling indexing of specific object types

By default, all instances of dm_sysobject or its subtypes are indexed. However, you can disable indexing of instances of specific object types in repositories at version 5.3 SP5 or later.

A type is registered for indexing if any of its supertypes are registered for indexing. You cannot turn off indexing for an object type if any of its supertypes are indexed. Because dm_sysobject and all of its subtypes are indexed (registered for indexing) by default, if you want to disable indexing for one or more subtypes, you must turn off indexing for dm_sysobject and then enable indexing specifically for those subtypes that you want indexed, leaving those you do not want to index as unregistered.

Use Documentum Administrator to disable or enable indexing. You must connect as a Superuser. The Properties page for each object type capable of being indexed has a checkbox named **Registered for indexing**. That checkbox indicates whether the object type is currently registered for indexing. The field is enabled or disabled depending on whether or not you can change the type's indexing status.

To remove or add registration for indexing for an object type:

1. Start Documentum Administrator as a Superuser.
2. Select the Types node.
3. Navigate to the object type whose registration you want to change and open its Properties page.

   This must be dm_sysobject or one of its subtypes.
4. To stop indexing of instances of the type, clear **Register for indexing**.
5. To resume indexing of instances of the type, select **Register for indexing**.

Configuring format objects to specify which renditions are indexed

Properties of the format object determine which formats are indexable and which content files in indexable formats are indexed. If the value of the can_index property of a content file's format object is set to TRUE, the content file is indexable. If the primary content of an object is not in an indexable format, you can ensure that the content file is indexed by creating a rendition in an indexable format.

The format_class property of the format object may be set to values that determine which formats are indexed:

- ft_always

  All renditions in formats whose format_class property is set to ft_always are indexed. For example, if a document has renditions in Microsoft Word and PDF formats and the format_class property for both formats is set to ft_always, both renditions are indexed.

- ft_preferred

  If a document has multiple renditions in indexable formats and one is in a format whose format_class property is set to ft_preferred, the rendition in that format is indexed rather than any renditions in other formats, with the exception that any formats whose format_class property is set to ft_always are also indexed. If a document has more than one rendition whose format_class property is set to ft_preferred, the first rendition processed for indexing is indexed and the other renditions are not. Which rendition is processed for indexing cannot be determined in advance. It is recommended that for any document, only one rendition is in a format whose format_class property is set to ft_preferred.

  If a document has renditions in four different formats, of which the format_class of one is set to ft_preferred and the format_class of the other three is set to ft_always, all four renditions are indexed.

There is no default value for the format_class property. You must set it manually to designate whether a format is always indexed or is the preferred format for indexing.

By default, the first content file in a format whose can_index property is set to true is indexed. Other renditions of the object are not indexed. If the primary content of an object is not in an indexable format, create a rendition in an indexable format. Appendix D, Supported and Unsupported Formats for Full-Text Indexing, contains a complete list of indexable formats.

If the content file associated with a SysObject exists in a non-indexable format, its properties are still indexed. To index the content, create a rendition of the SysObject in an indexable format. Use Documentum Content Transformation Services or third-party client applications to create the rendition.

Supported formats and mime_types

Appendix D, Supported and Unsupported Formats for Full-Text Indexing lists the formats considered indexable by the index server.

Some formats found in the appendix are not be represented in the repository by a format object. The formats.cvs file, which is located in $DM_HOME/install/tools, contains a complete list of supported mime_types and the formats with which they are associated. If a supported mime_type is not represented by a format object, create a format object in the repository and map the supported mime_type to the format.

Reindexing a repository

An existing index may become corrupted because of disk failure or corruption, host failure, or the unexpected shutdown of the index server. When the index is corrupted, it may be desirable to reindex the repository.

There are two ways to reindex a repository:

- Use an index agent running in migration mode to resubmit all qualifying repository objects for indexing.

  This is the recommended way to reindex. Refer to the *Content Server Full-Text Indexing Installation Guide* for complete instructions.

  Reindexing the repository rebuilds the existing index by replacing entries in the index. It is recommended that you run the index server in suspended mode if you reindex. For information on suspended mode, refer to Index server modes, page 122.
- Use the Create Full-Text Events tool (dm_FTCreateEvents) in full-reindex mode.

  This is not recommended, but if you do reindex with the tool, schedule it to run daily and process the default number of objects. Do not set the tool to run multiple times a day. The tool is described in the online help system for Documentum Administrator, from which it is run, and in Creating indexing events for new content in a repository, page 92.

Troubleshooting indexing timeouts

Occasionally, the processing time for a particularly complex file, for example, a complex spreadsheet, may be long and cause an indexing timeout. Use the procedure in this section if you are experiencing a timeout while indexing a file.

To modify index agent and index server timeout values:

1. Connect to the computer where the indexing software is installed as the installation owner.
2. Navigate to $DOCUMENTUM_SHARED/IndexAgents/IndexAgent1/webapps/IndexAgent1/WEB-INF/classes/ (UNIX and Linux) or %DOCUMENTUM_SHARED%\IndexAgents\IndexAgent1\webapps\IndexAgent1\WEB-INF\classes\.
3. Open the indexagent.xml file in a text editor.
4. In the <indexagent_instance> element, locate the <runaway_thread_timeout> and <runaway_item_timeout> elements.
5. Increase the default value.
6. In the <exporter> element, locate the <runaway_timeout> element.
7. Increase the default value.
8. In the <indexer> element, locate the <runaway_timeout> element.
9. Increase the default value.
10. In the <fast_indexer> element, locate the <fds_callback_wait_time> element.
11. Increase the default value.
12. Save the indexagent.xml file.
13. Stop the index server.

    Use the index agent admin tool.
14. Navigate to $DOCUMENTUM/fulltext/IndexServer/etc/processors/ (UNIX or Linux) or %DOCUMENTUM%\fulltext\IndexServer\etc\processors\
15. Open the Format.xml file in a text editor.
16. Locate the <load module="processorts.Format" class="StellentConverter"/> element.
17. Increase the timeout value to 3000:

    ```
    <param name="Timeout" value="3000" type="int"/>
    ```
18. Save the Format.xml file.
19. Navigate to $DOCUMENTUM/fulltext/IndexServer/etc/ (UNIX or Linux) or %DOCUMENTUM%\fulltext\IndexServer\etc\.
20. Open the NodeConf.xml file in a text editor.
21. In the Content Distributor section, ensure that the value of the --operation-timeout parameter is `6000`.

22. In the Config Server section, ensure that the last entry in the <parameters> element is `-p 12000`:

```
<parameters>-P $PORT -s no -M -ORBendPointNoListen
giop:tcp:pleearth:$PORT1
-ORBendPointNoPublish giop:tcp::$PORT1 -p 1200<parameters>
```

23. Save the NodeConf.xml file.
24. Restart the index server.

Creating a new index

If you must delete an existing index and create a new index for a repository, use the following high-level instructions.

To create a new index:

1. Delete the existing full-text indexing software and index, using the instructions in Appendix B, Uninstalling the Index Agent and Index Server.

   Ensure that you delete the existing index.
2. Reinstall the software, creating an index agent in migration mode, and reindex the repository, using the instructions in Chapter 7, Creating and Managing the Full-Text Index.

Pointing a repository to a previously-created index

If an index is created with the index agent in normal mode, the repository, index agent, and index server are automatically associated with one another.

If an index is created with the index agent in migration mode, a normal mode index agent must be created to update the index. If the repository is version 5.2.5.x, upgrade the repository and then configure a normal mode index agent. If the repository is version 5.3 or later, configure the normal mode index agent. Configuring the normal mode index agent associates the indexing software components, the index, and the repository.

Configuring index routing

To configure index routing, you must direct documents to particular storage areas and make configuration changes to the index agent and index server. The following sections discuss how to configure index routing. (For a conceptual description of index routing, refer to Multinode configuration with index routing, page 32.)

Directing documents to particular storage areas

To direct documents to particular storage areas, customize your applications to set the a_storage_type attribute for each document. For example, you might designate filestore_01 for PDF documents, filestore_02 for Word documents, and filestore_03 for Excel documents.

Configuring the index server

On the node where the ConfigServer process is running, the NodeConf.xml file must be modified to include the routing.cfg file and a routing.cft file must be created.

To configure the index server:

1. Shut down the index server.
2. Connect as the installation owner to the node where the ConfigServer process is running.
3. Navigate to the %DOCUMENTUM%\fulltext\IndexServer\etc ($DOCUMENTUM/fulltext/IndexServer/etc) directory.
4. Open the NodeConf.xml file in a text editor.
5. Add a collection routing definition to the status server entry:

```
<executable>statusserver</executable>
<parameters>>--type=single --db-dir=$FASTSEARCH/data/status
        --db-mem=1 -ORBendPointNoListen giop:tcp:host1:$PORT -ORBendPointNoPublish
        giop:tcp::$PORT --collection-routing-file=$FASTSEARCH/etc/routing.cfg
```

6. Save the NodeConf.xml file.
7. In the $DOCUMENTUM/fulltext/IndexServer/etc directory (UNIX) or%DOCUMENTUM%\fulltext\IndexServer\etc (Windows), create a file called routing.cfg.
8. Enter the collection to column mapping in the following format:

```
collectiondesignator=column_designator
```

For example:

```
collectiona=0
collectionb=1
collectionc=2
```

9. Save the routing.cfg file.
10. Restart all index server processes on all nodes.
11. Use the nctrl command on each node to verify that the processes are running correctly:

```
$ cd $FASTSEARCH/bin
 $ .  .setupenv.sh
$ nctrl  sysstatus
```

A list of index server processes and their status is displayed.

Configuring the index agent

On the index agent host, the indexagent.xml file must be modified to map the filestores to particular collections. The <partition_config> element is added immediately before the end of the <indexer> element.

In the following example, the default collection is defined and three storage areas are mapped to particular collections. Note that the collection to which filestore_01 is mapped is also defined as the default collection:

```
<partition_config>
    <default_partition>
         <collection_name>repb01</collection_name>
    </default_partition
    <partition>
          <storage_name>filestore_01</storage_name>
          <collection_name>repb01</collection_name>
    </partition>
    <partition>
          <storage_name>filestore_02</storage_name>
          <collection_name>repb02</collection_name>
    </partition>
<partition>
          <storage_name>filestore_03</storage_name>
          <collection_name>repb03</collection_name>
    </partition>
</partition_config>
```

```
</indexer>
```

Chapter 8

Managing Full-Text Indexing Components

This chapter contains information on starting and stop the index agent and index server, disabling index agents, viewing index server properties, and viewing or modifying index agent properties. You must have System Administrator or Superuser privileges to perform these tasks.

Topics in this chapter are:

Administration tools

Two tools are available for administering different aspects of full-text indexing:

- The Index Agent Admin Tool is installed when the index agent and index server are installed. The tool is a JSP page accessible at the URL

  `http://`*`hostname`*`:`*`portno`*`/IndexAgent`*`N`*`/login.jsp`

  where *hostname* is the name of the host where the index agent is running, *portno* is the port where the index agent is listening, and *N* is the number designating a particular index agent instance.

  Use the Index Agent Admin tool to map file stores, monitor the indexing process, and stop or start the index agent and index server on a particular host.
- Documentum Administrator

Use Documentum Administrator to stop or start index agents and index servers, to manage the full-text indexing queue, and to view index server logs.

Note: Using console mode in RemoteDesktop for the administration of the index server is not supported.

If the repository has a high-availability indexing configuration running, in which multiple, redundant index agents and index servers are installed and multiple, redundant indexes are maintained, the paired index agents and index servers are displayed together in Documentum Administrator, listed under the name of the index. The default names are:

- For the first index, *repositoryname*_ftindex_01
- For the first index server, FAST Fulltext Engine Configuration
- For the first and all subsequent index agents, *hostname*_IndexAgent1
- For the next index, *repositoryname*_ftindex_01_ha
- For the next index server, FAST Fulltext Engine Configuration - 2

Starting and stopping the full-text indexing system

Start the software components in this order:

1. Start the Content Servers.
2. Start the index server.
3. Start the index agent.

By default, at server startup, Content Server checks whether the index agent associated with the repository is started. If the index agent is already running, it remains running. If the index agent is not running and the start_index_agents parameter in the server.ini file is set to TRUE (the default value), the server starts the index agent. If the application server instance in which the index agent runs is not running, Content Server cannot start the index agent.

If the Content Server is shut down while the index agent is running, the index agent continues to run, waiting for Content Server.

Shut down the software in this order:

1. Shut down the index agent.
2. Shut down the index server.
3. Shut down the Content Servers.

If the software is stopped and started in the correct order, indexing continues normally after a restart, regardless of whether the index agent is in migration mode or normal mode. If the index agent is in migration mode, however, it is recommended that you monitor the position in the list of object IDs of the high-water mark queue item to ensure that indexing has not completed. A migration-mode index agent that is started against a repository in which indexing is completed may reindex the entire repository. The high-water mark is queue item with the object name "Full-text re-index high water mark" and it traverses the repository in ascending order of object IDs.

Starting and stopping the index agent

Use these instructions to stop a running index agent or start an index agent that is stopped. When the index agent is running in normal mode, you can use Documentum Administrator or the Index Agent Admin Tool to start and stop it. When the index agent is running in migration mode, use the Index Agent Admin Tool to start or stop it.

Stop the index agent before you stop its associated index server. If you try to stop the index server first, Documentum Administrator automatically stops the index agent as well. The JSP interface warns you to stop the index agent before you stop the index server.

If the index server is stopped and you try to start its associated index agent, you are asked whether you want to also start the index server. If the status of the index server is **Not Responding**, you cannot start or stop the index server, but you can start or stop the associated index agent.

An index agent that is disabled cannot be started and is not started automatically when its associated Content Server is started. You must enable the index agent first. For information on enabling a disabled index agent, refer to Enabling and disabling index agents, page 107. If the index agent's status is **Not Responding**, examine the machine on which it is installed and ensure that the software is running.

Note that stopping or starting an index agent does not stop or start the application server process in which the index agent runs. You must stop the application server processes manually. However, if you stop an application server process, you must restart it before the index agent can be started.

Caution: Stopping the index agent interrupts full-text indexing operations, including updates to the index and queries to the index. An index agent that is stopped does not pick up index queue items or process documents for indexing.

To start or stop an index agent using Documentum Administrator:

1. Log into Documentum Administrator, connecting to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Agents and Index Servers**.

3. Select the correct index agent.
4. To start the index agent, click **Tools>Start**.
5. To stop the index agent, click **Tools>Stop**.
6. Confirm that you want the index agent started or stopped.
   The index agent's status changes to running or stopped.

To start or stop an index agent using the Index Agent Admin Tool:

1. Log in to the Index Agent Admin Tool.
   The Index Agent Admin Tool is available at this URL:
   `http://hostname:portno/IndexAgentN/login.jsp`
   where *hostname* is the name of the host where the index agent is running and *portno* is the port where the index agent is listening and *N* is the number assigned to the index agent instance. If the browser is on the index agent host, replace *hostname* with `localhost`.
2. To start the index agent, click **Start** in the index agent status line.
3. Click **Ok**.
4. To stop the index agent, click **Stop** in the index agent status line.
5. Click **Ok**.

The dm_FTIndexAgentBoot job

The dm_FTIndexAgentBoot job is installed when you install the full-text indexing components. The job is set to the active state when the index agent is configured. The job periodically checks the state of the index agent and if the agent is not running, restarts the index agent.

Starting and stopping the index server

Use these instructions to stop a running index server or start an index server that is stopped. You can use Documentum Administrator or the Index Agent Admin Tool to start and stop it. If you stop an index server, its associate index agent is also stopped, and you are informed that the index agent will be stopped as well.

If the index server's status is **Not Responding**, attempt to retrieve and examine the log files, using the instructions in Reviewing the index agent and index server log files, page 109. If you cannot retrieve the log files, examine the machine on which the index server is installed and determine whether it is running.

When you stop the index server, wait a few minutes before attempting to restart it. Note that stopping or starting an index agent does not stop or start the application server process in which the index server runs.

Caution: Stopping the index server interrupts full-text indexing operations, including updates to the index and queries to the index.

To start or stop an index server using Documentum Administrator:

1. Log into Documentum Administrator, connecting to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Agents and Index Servers**.
3. Select the correct index server.
4. To start the index server, click **Tools>Start**.
5. To stop the index server, click **Tools>Stop**.
6. Confirm that you want the index server started or stopped.

   The index server's status changes to running or stopped.

To start or stop an index server using the Index Agent Admin Tool:

1. Log in to the Index Agent Admin Tool.

   The Index Agent Admin Tool is available at this URL:

   `http://`*`hostname`*`:`*`portno`*`/IndexAgent`*`N`*`/login.jsp`

   where *hostname* is the name of the host where the index agent is running and *portno* is the port where the index agent is listening and *N* is the number assigned to the index agent instance. If the browser is on the index agent host, replace *hostname* with `localhost`.
2. To start the index server, click **Start** in the index server status line.
3. Click **Ok**.
4. To stop the index server, click **Stop** in the index server status line.
5. Click **Ok**.

Enabling and disabling index agents

An index agent that is disabled cannot be started and is not started automatically when its associated Content Server is started. You can disable an index agent whether it is running or stopped. To start a disabled index agent that is not running, you must enable the index agent first.

You may wish to disable an index agent if the computer on which the index server is installed has had a hardware failure or if the index agent itself has failed.

To enable or disable an index agent:

1. Log into Documentum Administrator, connecting to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Agents and Index Servers**.
3. Select the correct index agent.
4. Click **Tools>Enable** or**Tools>Disable**.
5. Confirm that you want the index agent enabled or disabled.
   The index agent's status changes to the selected state.
6. If you enabled the index agent, restart it.

Viewing or modifying index agent properties

Use these instructions to view the properties of an index agent. You can modify the following index agent properties, but it is recommended that you do not change the values:

- Exporter Thread Count

  This is the number of concurrent exporter threads run by the index agent. The default value is 3. If you change the exporter thread count, you must restart the index agent for the change to take effect.
- Polling Interval

  This is the frequency, in seconds, at which the index agent polls for queue items. The default value is 60.

All other properties are read-only.

To view or modify index agent properties:

1. Log in to Documentum Administrator, connecting to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Agents and Index Servers**.
3. Click the Info icon for the index agent.
4. If required, modify the exporter thread count or polling interval properties.
   It is recommended that you do not modify the default values.
5. Click **Ok** to save the changes or **Cancel** to exit without saving.

Viewing index server properties

You cannot modify the properties of an index server. To view the properties of an index server, click the **Info** icon next to the name of the server in Documentum Administrator (**Administration>Indexing Management>Index Agents and Index Servers**). The following properties are displayed:

- The index server name

  This is the object name of the configuration object.
- The name of the host on which the index server is running
- The base port number used by the index server

  The default value is 13000.
- Collection name

  The name of an index collection, which typically contains the index data for a particular repository.
- Description

  A description of the collection, including the repository for which the collection contains the index data.
- Cluster

  The search cluster to which the collection belongs. The default name is webcluster.
- Pipeline

  The document processing pipeline for a particular collection. The default value is DFTXML.

For 5.3 repositories with the indexing software installed, only one row is displayed because a 5.3 index server managed only one collection. In 5.3 SP1 and later repositories, an index server can manage multiple collections.

Reviewing the index agent and index server log files

The index agent log file is located in $DOCUMENTUM_SHARED/Logs on UNIX or Linux or %DOCUMENTUM_SHARED%\Logs (Windows). The file name is IndexAgent*N*.log, where *N* is the number of the particular index agent. For example, in a consolidated indexing deployment, in which more than one repository is indexed by a single index server, you may have IndexAgent1, IndexAgent2, and IndexAgent3 running, with each corresponding to a particular repository. The log file for IndexAgent3 is called IndexAgent3.log.

To view the logs for an index server, use Documentum Administrator.

To view the logs of an index server:

1. Log into Documentum Administrator, connecting to the repository as user who has System Administrator or Superuser privileges.
2. Click **Administration>Indexing Management>Index Agents and Index Servers**.
3. Select an index server.
4. Click **View>Get Index Server Logs**.
   A zip file containing the logs is produced.
5. Save the zip file to the local drive.
6. Unzip the compressed file and view the logs.

Administration operations

Use the instructions in this section to perform configuration and administrative tasks.

Note: Using console mode in RemoteDesktop for the administration of the index server is not supported.

Configuring batched returns for non-FTDQL queries

Queries that contain a SEARCH clause but are not FTDQL queries are processed using a temporary table. Content Server populates a temporary table with the results returned by the SEARCH clause and then filters the results for security and matches to any other conditions specified in the query. If the SEARCH clause has returned a large number of results, processing them may take some time. To improve performance, the results are populated into the temporary table and, with some exceptions (refer to the Note) are processed in batches. The batch size is configurable.

The batch size is controlled by the temp_table_batch_size parameter for the full-text engine configuration. This parameter is set in the dm_ftengine_config object. The parameter name is set in param_name property and the value is set in the param_value property. These are repeating properties, so you must set the name and value at the same index position within the property. The value is an integer number representing the number of results in each batch.

You must reinitialize Content Server after setting the parameter.

If this parameter is not set, the default batch size is 20000. Setting the parameter to 0 disables the batching feature.

Note: Batched returns are not used if the query contains any of the following:

- An aggregate function, such as count() or sum().
- A UNION, IN DOCUMENT, IN ASSEMBLY, or ORDER BY clause
- A SEARCH clause in a subquery

In such cases, all results are populated into the temporary table for processing in one batch.

Because the full-text engine may return duplicate hits for an object, Content Server also processes the full-text results within each batch to remove duplicates. Note however, that if there are duplicate rows across the batches, those duplicates are not removed.

Configuring duplicate checking batch size

If the search results are populated into the temporary table in batches, Content Server removes any duplicates within each batch. If all results are populated into the table in one batch, Content Server removes any duplicates in the set, unless the set exceeds the size limit specified in the temp_table_remove_dup_size parameter set in the dm_ftengine_config object. If the size of the result set exceeds that parameter, Content Server does not attempt to remove duplicates. The parameter's default is 20000. You can change the default.

The temp_table_remove_dup_size parameter is set in the dm_ftengine_config object. The parameter name is set in param_name property and the value is set in the param_value property. These are repeating properties, so you must set the name and value at the same index position within the property. The value is an integer number representing the number of results in each batch.

Setting the parameter to 0 disables this feature.

You must reinitialize Content Server after setting the parameter.

Enabling thesaurus searching

Thesaurus searching allows users to define which words are returned by a particular search.

A synonym file contains the definitions. Typically, entries in the file contain words with similar meanings (synonyms). For example, you can define "auto" and "automobile" as synonym entries that must be returned when the word "car" is the search term. However, an entry can contain unrelated words or create user-defined associations.

A synonym file may also containing spelling variations and abbreviations. For example, "favour" is the British spelling of "favor" and "fvr" is an abbreviation of "favor."

Thesaurus searching is supported for only one language per index server installation. All languages listed in Appendix E, Supported Languages for Full-Text Indexing, are supported for thesaurus searching. The default language for thesaurus searching is English.

This section contains instructions for creating a synonym file and importing its contents to create a synonym dictionary. Once the synonym file is imported and the synonym dictionary is created, thesaurus searching is enabled. User searches return not only the searched-for word, but also any synonyms defined in the synonym file.

If you upgraded from an EMC Documentum version prior to 5.3, an existing Verity thesaurus file can also be used to create the synonym dictionary.

You can enable thesaurus searching at any time. You can also add entries to an existing synonym dictionary.

Creating the synonym file

The synonym file defines which words you want to be returned by a particular search. Create a text file with synonym entries in the following format:

term=[[*spelling variations*],[*abbreviations*],[[*synonyms*][]]]

The file may have any name. Save the file in the UTF-8 code page.

The brackets in each entry are literal characters that separate spelling variations, abbreviations, and synonyms. The empty brackets at the end of the synonym entry are required.

The following examples contain only synonyms::

```
car=[[],[],[[automobiles,autos],[]]]
cars=[[],[],[[automobiles,autos],[]]]
auto=[[],[],[[automobiles,autos],[]]]
autos=[[],[],[[automobiles,autos],[]]]
```

The following example contains a spelling variation, an abbreviation, and synonyms:

```
favor=[[favour],[fvr],[[prefer,privilege],[]]]
```

There are no limitations on the length of an entry or the number of entries in the synonym file.

Importing the synonym file

You can import a synonym file created as described above in Creating the synonym file, page 112 or you can import an existing Verity thesaurus file.

To import the synonym file, connect to the index server host as the index server installation owner and run the ImportDictionary.py script. The script is executed by Cobra, which is included in the index server installation. The following conditions must be met to run ImportDictionary.py:

- The FASTSEARCH environment variable must be set ($FASTSEARCH on UNIX, %FASTSEARCH% on Windows).
- The script must be run from the $FASTSEARCH/bin directory (UNIX) or %FASTSEARCH%\bin folder (Windows).
- The synonym file must be found in the $FASTSEARCH/bin directory (UNIX) or %FASTSEARCH%\bin folder (Windows).

The syntax is

```
cobra ImportDictionary.py [-I filename -M verity|fast -L
language -T transaction_type]
```

or

```
cobra ImportDictionary.py [--importfile=filename --mode=verity|fast
--language=language --transactiontype=transaction_type]
```

The arguments are described in Table 8–1, page 113.

Table 8-1. Syntax of ImportDictionary.py script

Argument	Description
-I *filename*or --importfile *filename*	The name of the synonym file. May be any arbitrary name.
-M *verity*|*fast* or --mode*verity*|*fast*	Whether an existing Verity thesaurus file or a new index server synonym file is being imported. Allowable values are verity and fast. If the argument is not included, the default value is fast.

Argument	Description
-L or --language	The language of the synonym file. Allowable values are the names of the languages listed in Appendix E, Supported Languages for Full-Text Indexing. If the argument is not included, the default value is English.
-T *transaction_type* or --transactiontype *transaction_type*	The allowable values are: • createnew, which instructs the index server to create a new synonym dictionary using the terms in the file *filename*. • insertentries, which instructs the index server to add entries from the synonym file *filename* to an existing synonym dictionary. • deletedictionary, which instructs the index server to delete the existing dictionary. Do not provide a file name if you run ImportDictionary.py to delete an existing synonym dictionary. If you do not provide an argument, the English dictionary is deleted by default.

For example, to import a new file called MySynonyms.txt:

```
cobra ImportDictionary.py -I MySynonyms.txt -M fast -L English -T createnew
```

To use an existing Verity thesaurus file called VeritySynonyms:

```
cobra ImportDictionary.py -I VeritySynonyms -M verity -L English -T createnew
```

To add new synonyms from a file MoreSynonyms.txt to an existing synonym dictionary:

```
cobra ImportDictionary.py -I MoreSynonyms.txt -M fast -L English -T insertentries
```

To delete an existing synonym dictionary:

```
cobra ImportDictionary.py -T deletedictionary
```

Logging

Running the ImportDictionary.py script generates a log file in $FASTSEARCH/bin (UNIX) or %FASTSEARCH%\bin (Windows). The log file is called error.log. It is generated each time the ImportDictionary.py script is run, whether or not errors are reported. The information in the log file is also displayed on your console or monitor while the script is running. The log file contains information on:

- Which options were used to run the script
- The name of the synonym file
- Whether required files are present in the index server installation
- The actions performed by the script
- When any errors occurred

- How to correct the errors
- The name of the synonym dictionary that is created
- Which index server processes create the synonym dictionary

Log sample

The following is a sample log:

```
Information : Logging started
Services ['storageservice', 'j2ee'] will be started
IMPORTANT!!!:The current File C:\Documentum\fulltext\IndexServer\etc\NodeConf.xml
is being modified as C:\Documentum\fulltext\IndexServer\etc\NodeConf.xml.origRestore
it back before rerunning the script if script fails Launching File Transformer
C:\Documentum\fulltext\IndexServer\bin\cobra with args ['convert.py',
'veritysyn.txt', 'FASTFile', 'en']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\cobra
The argument list is ['cobra', 'convert.py', 'veritysyn.txt', 'FASTFile', 'en']
Successfully generated the transformed file FASTFile
Copying NodeConf.xml.mod as C:\Documentum\fulltext\IndexServer\etc\NodeConf.xml
to start services


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'reloadcfg']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'start', 'storageservice', 'j2ee']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'sysstatus']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'sysstatus']
The dictionary app is C:\Documentum\fulltext\IndexServer\bin\dictman
The documentum_synonym_en dictionary will be generated
The DictmanCmdFile command file will be used by dictman
The Dictionary documentum_synonym_en  exists
The importFile FASTFile
```

```
############################################################
Starting Dictionary documentum_synonym_en import
Creating Dictman command file DictmanCmdFile


############################################################

Information : File DictmanCmdFile created
Writing strings
 open /documentum_synonym_en
thesimport /documentum_synonym_en FASTFile
close /documentum_synonym_en
compile /documentum_synonym_en
quit

Generated DictmanCmdFile command file for Dictman


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\dictman
The argument list is ['dictman', '-u', 'root', '-f', 'DictmanCmdFile']
imported entries successfully into Dictionary documentum_synonym_en
Stopping ['qrserver', 'search-1'] services


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'start', 'qrserver', 'search-1']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'sysstatus']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'stop', 'qrserver', 'search-1']


############################################################

Information : Launching application C:\Documentum\fulltext\IndexServer\bin\nctrl
The argument list is ['nctrl', 'stop', 'qrserver', 'search-1']
Error : Application C:\Documentum\fulltext\IndexServer\bin\nctrl returned
with error code -5
Error : Launch failed for application C:\Documentum\fulltext\IndexServer\bin\nctrl
Error : Application C:\Documentum\fulltext\IndexServer\bin\nctrl returned
with error code -5
```

Obtaining a list of indexable formats

Use the following DQL query to obtain a list of content formats that are marked as indexable:

```
SELECT "name", "description" FROM "dm_format"
WHERE "can_index"=true
```

Tracing full-text query operations

Use the MODIFY_TRACE administration method to turn on tracing for full-text querying operations. The method is executed from Documentum Administrator. Two tracing levels are available: None, in which tracing is turned off, and All, in which all Content Server and full-text messages resulting from queries are logged. The fttrace_*repository*. log file contains messages generated by turning tracing on using MODIFY_TRACE.

When tracing is turned on, FTDQL queries are logged with the following format:

```
FTDQL Execution. ft_convertible = T|F, converted_ft_query
= %s, ft_search_type = %s
```

When ft_convertible is TRUE, converted_ft_query contains the converted full-text equivalent of the WHERE clause. If a where clause was not specified in the query's select statement, converted_ft_query is empty. The ft_search_type indicates how the query was executed against the index.

When tracing is turned on and non-FTDQL queries are executed, the SQL statements created for the DQL queries are logged to the full-text log file.

Enabling tracing for the index agent

Use these instructions to enable tracing for the index agent.

To enable tracing for the index agent:

1. On the index agent host, navigate to the $DOCUMENTUM_SHARED/config directory (%DOCUMENTUM%\config on Windows).
2. Open the IndexAgent*N*.log4j.properties file, where *N* is the number of the index agent for which you are enabling tracing.
3. Locate the following text:

   ```
   log4j.category.com.documentum=INFO
   ```

4. Replace `INFO` with `DEBUG`:

```
log4j.category.com.documentum=DEBUG
```

5. Save the file.

If a node fails in a high-availability configuration

In a high-availability configuration, one index is defined as the default index and the other index is defined as the standby index. The value of the is_standby attribute of the fulltext index objects determines which is the default index. If is_standby is set to FALSE, the index represented by the fulltext object is the default index. If is_standby is set to TRUE, the index is the standby index.

If a node fails, indexing operations continue for the remaining index. If the remaining index is the default index, querying operations continue. If the remaining index is the standby index, querying operations stop. Use Documentum Administrator to change the value of the is_standby attribute of the two fulltext index objects:

- Change the default index to the standby index by setting the value to TRUE
- Change the standby index to the default index by setting the value to FALSE

You must restart any running Content Servers after setting the values. Queries are then directed to the index that is active.

Cleaning up old log files

Periodically, the indexing system automatically archives the all.log file. The archived file is stored as a zip file in the C:\Documentum\fulltext\IndexServer\var\log\Archive (Windows) or $DOCUMENTUM/fulltext/IndexServer/var/log/Archive (UNIX) directory. These archived files are not removed automatically. They remain in the archive directory until they are manually removed.

Large file rejection error

The fulltext engine is configured to reject for indexing any FIXML file larger than 10MB. The rejection is logged in the dmi_queue_item and in the index agent log file. The error message is:

```
DOCUMENT_ERROR Module :FIXMLFilter - Error: Process(objectID)
failure: docproc.ProcessorStatus.NotPassing'
```

where *objectID* is the object ID of the document containing the rejected content.

Increasing capacity

Use the information in this section if you require increased indexing or querying capacity.

Increasing indexing capacity

If the primary requirement is to increase indexing capacity — the ability to index a greater number of documents in a particular time period — several strategies are available:

- Adding index server nodes — using a multinode configuration — increases indexing capacity.

  In multinode configurations, documents are routed sequentially to different nodes, so that a particular node handles a fraction of the indexing. In a four-node configuration, each node will index about one-fourth of the data.
- Add docprocessor (procserver) processes on each host.

  This increases the speed with which FIXML is processed. Instructions for adding docprocessor processes are included in the April, 2006 *Content Server Release Notes* for server version 5.3 SP2.
- Adding index agents

Increasing the number of exporter threads in the index agent

The number of index agent exporter threads is configurable. Increasing the number of threads increases the speed at which the index agent acquires queue items. The default value is 3. The value may be increased. It is recommended that the value does not exceed 12.

Chapter 9

Full-Text Indexing Components in Detail

Full-text indexing is supported for installations with repositories running on Microsoft Cluster Services provided the indexing software is installed on a host other than the Content Server host. This chapter contains the following topics:

The index server in detail

The index server has two functions: it creates full-text indexes and responds to full-text queries from Content Server.

An index server node is any physical host on which an index server instance runs, regardless of whether multiple instances of the index server's individual software process are running.

The index server is represented in the repository by the ft engine config object (dm_ftengine_config). There is one ft engine config object for each fulltext index object.

Index server processes

The index server consists of five groups of processes that have different functions. In a basic configuration, these processes all run on the same host. In a multinode configuration, these processes are run on different hosts to improve performance. The processes are:

- Administrative services

  The index server's administrative services are the *Status Server*, which monitors the progress of a document through the index server's internal processes from when the document is acquired from the index agent to the time the document is indexed, and the *Content Distributor*, which accepts DFTXML from the index agent and routes it to a document processor.

- Document processors

  Document processors (also sometimes called *procservers*) extract indexable content from content files, convert DFTXML to FIXML (a format that is used directly by the indexer), and merge the indexable content with the metadata during the DFTXML conversion process. Document processors are the largest consumer of CPU power in the index server. In multinode configurations, a particular document processor communicates with the indexers on all other hosts in the configuration.

- Indexer

  The indexer creates the searchable full-text index from the intermediate FIXML format. It consists of two processes. The *frtsobj* process interfaces with the document processor and spawns different *findex* processes as necessary to build the index from FIXML.

- Query and results servers

  The *QR Server* (Query and Results Server) is a permanently-running process that accepts queries from Content Server, passes queries to the fsearch processes, and merges the results when there are multiple fsearch processes running. In a multinode configuration, the QR Server is installed on the node where administrative processes are running.

- Search servers

  Search servers locate items in the index as specified in a query. The fsearchctrl, fdispatch, and fsearch processes work together to fulfil requests from the qrserver process.

In a single-node deployment, all of these processes reside on one host.

In a multinode deployment, copies of the document processors, the Indexer, and the search servers reside on each node. The administrative processes and the query and results servers reside on only one of the nodes.

Index server modes

The index server can run in continuous mode or in a special mode called suspended mode. In suspended mode, FIXML is generated for any updates to the index but not

integrated into the index. When the index server is taken out of suspended mode, the index is updated.

Running in suspended mode speeds up the indexing process. Use suspended mode in these circumstances:

- When creating a new index as part of migrating from a pre-5.3 Content Server
- When reindexing a repository
- When indexing a large volume of documents

Use the index agent JSP page to put the index server in suspended mode.

The index agent

The index agent is a multi-threaded Java application running in the Apache Tomcat servlet container. Each index agent runs in its own Tomcat instance, and each index agent is associated with only one repository.

The index agent and the index server must be installed on the same host.

Certain operations on objects in the repository generate queue items indicating that the object and any associated content files must be submitted for full-text indexing. The index agent reads the queue items, then exports the content files and metadata from the repository and prepares the documents and metadata for indexing by converting them to DFTXML.

An index agent in normal mode is represented by an ft index agent config object. The properties of the ft index agent config object primarily record status information about the index agent, including the mode in which the index agent is running and when the index agent began processing queue items. The properties also record configuration information about the index agent, such as the number of queue items processed in a single batch, the number of exporter threads, and the time interval at which the index agent polls the repository for queue items. This information may be viewed using Documentum Administrator. For more information about the ft index agent config object, refer to the *EMC Documentum Object Reference Manual*.

An index agent in migration mode is represented by an XML configuration file, indexagent.xml, on the index agent host. Do not modify the parameters in the configuration file unless you are enabling file store mapping. (Mapping file stores for improved indexing performance is documented in Deciding whether to share the drives where content files are located, page 52.)

Index agent modes, page 125, contains complete information on the modes in which the index agent runs.

Index agent processes

When running in normal mode, the index agent processes index queue items and the SysObjects associated with the queue items. The index queue consists of dmi_queue_items that are queued to the dm_fulltext_index_user. The queue items are generated by Save, Saveasnew, Checkin, Destroy, or Branch events performed on a SysObject or SysObject subtype in the repository. The value of the queue item's task_state property changes depending on where the SysObject is in the indexing process:

1. The initial task_state is value is a blank, indicating that the associated SysObject must be indexed and the queue item is available for processing by the index agent.
2. When the index agent acquires the queue item and begins to process the SysObject, the task_state is updated to Acquired.
3. When the SysObject is successfully indexed by the index server, the task_state is updated to Done and the queue item is deleted from the repository.

   No further updates are made to the queue item.
4. If the index agent encounters a problem in processing the SysObject to which the queue item refers, the task_state is changed to Warning.

   This indicates that the index server has made a partial update to the index. The index agent does not further update the queue item or perform further operations on the SysObject. The queue item remains in the queue.
5. If indexing fails and the index server is unable to add information about the SysObject to the index, the task_state is updated to Failed.

   The index agent does no further processing on the SysObject and the queue item remains in the queue. To determine which objects failed indexing, use Documentum Administrator.

By default, the index agent acquires queue items in batches of 1,000. If the index agent shuts down unexpectedly after acquiring queue items, the queue items remain acquired by that index agent. When the index agent is restarted, the task_state property for those acquired queue items is reset to ' ' and the queue items are processed correctly.

If the index agent acquires multiple queue items for a particular SysObject, it processes only one of the queue items. For example, if an object is saved several times, the most recent version is indexed. If an object is saved and then deleted, only the delete is processed.

Managing the index queue, page 78, contains complete information on managing the index queue.

Index agent modes

An index agent may run in one of three operational modes:

- normal

  In normal mode, the index agent process index queue items and prepares the SysObjects associated with the queue items for indexing.

- migration

  In migration mode, the index agent processes all SysObjects in a repository sequentially in r_object_id order and prepares them for indexing. A special queue item, the high-water mark queue item, is used to mark the index agent's progress in the repository.

- file

  In file mode, a file is used to submit a list of objects IDs to the index agent when a new index is created and index verification determines which objects are missing from the index.

The index agent runs in either migration mode or normal mode against a 5.3 repository and in migration mode against a 5.2.5 repository. Both modes are compatible with file mode, which is used for submitting a list of object IDs to the index agent for indexing.

Normal mode

Content Servers 5.3 and later generate a queue item when an event such as a check-in or save requires that a new or modified object must be indexed. In normal mode, the index agent reads the queue item, prepares the object for indexing, and updates the queue item. When the index agent successfully submits the object for indexing, the index agent deletes the queue item from the repository. If the object is not submitted successfully, the queue item remains in the repository and the error or warning generated by the attempt to index the object is stored in the queue item. The index agent can run in normal mode only against a 5.3 or later repository.

An index agent in normal mode and an index agent in migration mode cannot simultaneously update the same index.

Migration mode

In migration mode, the index agent prepares all indexable objects in a repository for indexing in object ID order. A special queue item, the high-water mark, records the ID of the most recent object indexed. The index agent reads the value in the queue item,

exports the next batch of indexable objects from the repository, and updates the queue item. The index agent can run in migration mode to create new indexes against a 5.2.5.*x*, or 5.3.*x* repository.

For example, if you need to move the indexing component installation to a new host computer, you can install the components on the new computer and create a new index there in migration mode, while the existing indexing component installation maintains the existing index in normal mode.

In migration mode, the index agent prepares all indexable objects for indexing in object ID order. A single queue item records the ID of the most recent object indexed.

File mode

File mode is used for submitting a list of object IDs to the index agent for indexing.

File mode can be used when the index agent is in any operational mode. However, it is most useful immediately after a new index is created. After a new index is created, the ftintegrity tool is run to determine whether all indexable objects in the repository are included in the index. The ftintegrity tool produces a list of object IDs of SysObjects that are not included in the index. The list is used to submit the objects to the index agent for indexing.

The full-text index

An index is made up of nodes. Depending on your deployment configuration, an index may have a single node or multiple nodes.

Each node has one column. A column is a physical set of processes and partitions that contain a portion of the indexed data. Each column has three partitions and the processes that search those partitions.

The data in each column is unique. That is, a document's indexed data resides on only one node. A column may contain data from one repository or multiple repositories.

Partitions

A partition is the smallest physical grouping of documents within a column. By default, each column has three partitions: a small partition, a medium partition, and a large partition. The documents most recently submitted for indexing are initially indexed into the small partition. The index server moves index data from the small partition to the

medium partition, and finally to the large partition. The index data for all documents eventually is in the large partition.

While new documents are being added to a partition, the partition is offline and in the process of being rebuilt. New documents are not added to the index while the partition is offline. A copy of the partition remains online and searchable. When the offline partition is completely rebuilt, it becomes online and searchable, and the online copy is taken offline and newly-processed FIXML is indexed.

Collections

All documents in an index belong to a collection, a logical set of data. The full-text indexing engine uses collections to segregate data logically. If the index stores data from only one repository in a single-node deployment, the data is typically in one collection.

In consolidated deployments, in which multiple repositories are indexed in the same index, the indexed data from each repository is associated with a separate collection.

In a multi-node deployment, a particular collection may be spread out over multiple nodes or it may be stored on only one node. The default behavior in a multinode deployment is to store the collections across nodes. However, if you wish, you can configure the system to use directed routing. This behavior requires you to store content files in particular storage areas, which are mapped to particular collections, and in turn, those collections are mapped to particular nodes. Which content files go to which storage area is a business decision.

Directed routing

Directed routing is the term that refers to the ability to direct index data to particular nodes. Using a multinode deployment with directed routing is recommended if you need to separately back up and restore individual collections or if you have a very large repository (in this release, 20 million or more documents).

Directed routing is implemented by mapping each file store storage area to a particular collection, and then mapping each collection to a particular column. Any arbitrary number of storage areas may be mapped to a particular collection. The mapping may be modified after installation to add more collections, though the index agent must be restarted after the mapping is modified.

Directed routing should only be used in the following circumstances:

- If some large amount of the data will be unchanging (never updated) and you wish to reduce the amount of data to index.

- If the full-text indexing system is growing and you need to incrementally add hardware to the system with little or no need to reorganize the indexed data.

  Directed routing allows you to fill a node and move on to newly added nodes. If you were not using directed routing in these situations, the system would perform additional work to redistribute the data across all of the new nodes.

For information about setting up directed routing, refer to Configuring index routing, page 100.

Repository objects and properties supporting full-text indexing

Full-text indexing is supported in the repository by objects representing the full-text index, the index server, and the index agent. Properties of the server config object, SysObject, and queue item also support full-text indexing.

Full-text indexing is supported in the repository by the following objects:

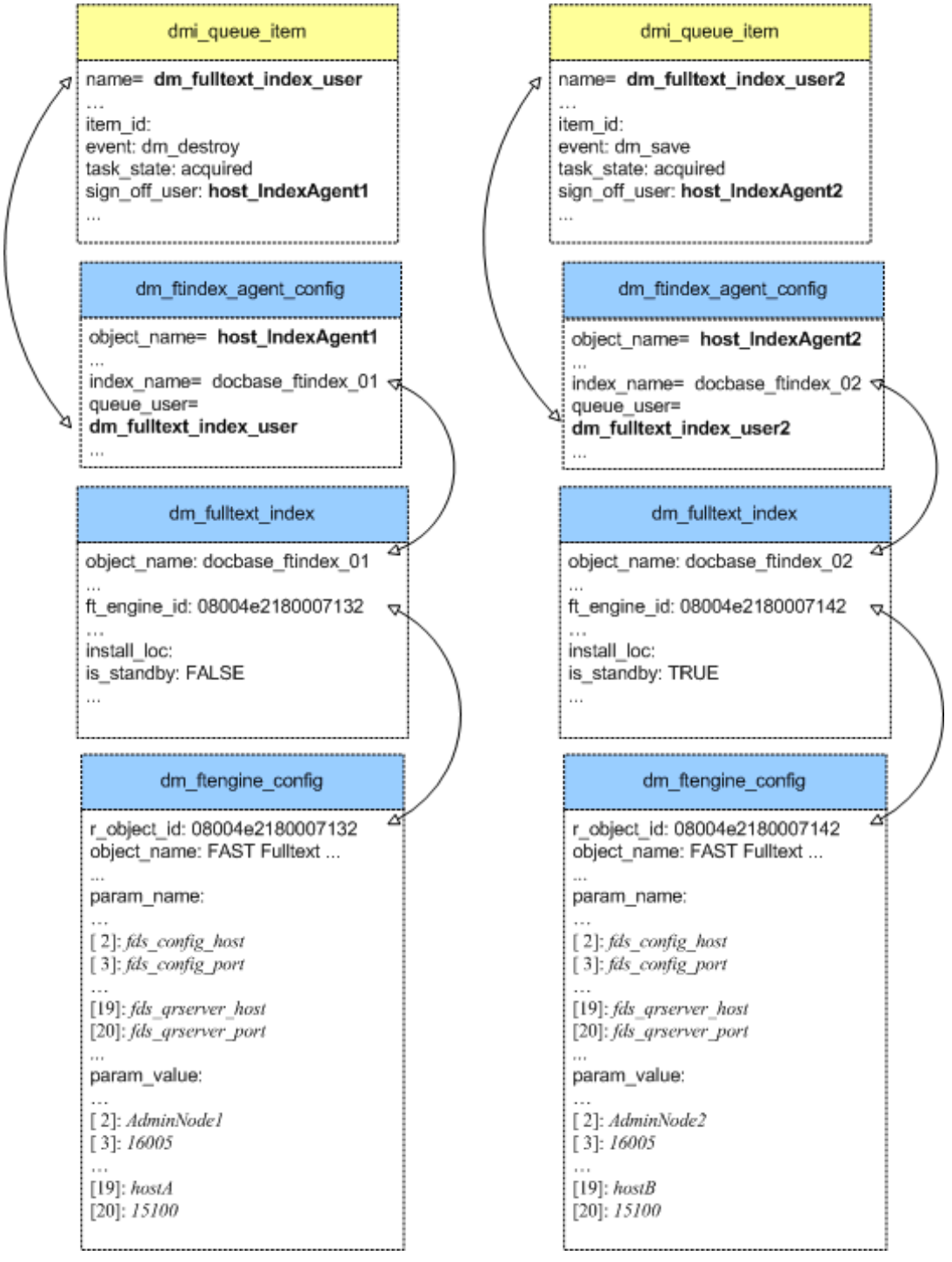
- fulltext index object, representing the full-text index
- ft index agent config object, representing the index agent
- ft engine config object, representing the index server

The fulltext index object is created at repository configuration time. In a basic full-text indexing configuration, the ft index agent config object and ft engine config object are created in the repository when the index agent configuration program creates a normal-mode index agent

In a high-availability configuration, a fulltext index object, ft index agent config object, and ft engine config object must be created to represent the second indexing software installation and the index maintained by the second installation. Certain attribute values must be set in the objects created to ensure that the index is updated correctly. In addition, a second full-text index user (dm_fulltext_index_user_01) must be created and the Save, Checkin, Destroy, Readonlysave, and MoveContent events must be registered for that user. A script is provided to create the required objects, set the attribute values, and register the events for the second full-text index user.

Figure 9–1, page 129, shows the objects involved and the relationships among them:

Figure 9-1. Full-text indexing object relationships



Refer to Installing a high-availability deployment, page 64, for high-level information on installing a high-availability configuration and on the objects that must be created and attribute values updated after installation.

Fulltext index object

A fulltext index object (dm_fulltext_index) represents an index associated with a repository. Its properties provide information such as the code page in which the Content Server represents strings sent to the query plug-in, the index name, the object ID of the index server for the index, and the name of the location object containing the location of the query plug-in. A fulltext index object is created when the first normal mode index agent is configured for the repository. The only property of the fulltext index object that may be updated is the ft_engine_id property, which contains the object ID of the ft engine config object representing the index server.

FT index agent config object

The ft index agent config object represents a normal-mode index agent configured for the repository. Its properties record status information and configuration information about the index agent.

FT engine config object

The ft engine config object represents the index server for the repository.

Location objects

Each fulltext index object points to a location object that represents the location of the dmfulltext.ini file.

Supporting properties of other objects

The following properties of other objects support full-text indexing:

- a_full_text

- fulltext_location

Configuring format objects to specify which renditions are indexed, page 96, contains information on how the setting of the format_class property of the format object controls which renditions are indexed.

The a_full_text property

The a_full_text property is defined for the SysObject type and is inherited by all SysObject subtypes. It is a Boolean property that controls whether an object's associated content files are indexed.

When a_full_text is TRUE, content files are indexed whenever a Save, Checkin, Destroy, Readonlysave, or MoveContent operation generates an index queue item for the object. Any changes to the object's content are added to the index.

The a_full_text property is set to TRUE whenever a SysObject is created. Users with Sysadmin or Superuser privileges can change the a_full_text setting. Users without Sysadmin or Superuser privileges cannot change the a_full_text setting.

The fulltext_location property

The value of the fulltext_location property of the server config object contains the name of the location object that identifies the fulltext configuration file, dmfulltext.ini.

Initialization files

This section describes the entries in the server initialization file that support full-text indexing and the dedicated fulltext initialization file.

Full-text entries in the server.ini file

Two parameters in the server.ini file govern aspects of full-text indexing:

- max_ftacl_cache_size

  Content Server caches ACL information on objects to evaluate security on the results returned by full-text queries. The max_ftacl_cache_size key defines the number of elements cached.

The default value is -1 (no limit set). If the value is set to 0, no security information is cached. The value may be set to any integer greater than -1. It is strongly recommended that you do not change the default value.

- start_index_agents

  The start_index_agents parameter defines whether the index agents associated with a particular repository are started at Content Server startup. The default value is TRUE. Changing the value to FALSE means that no index agent is available to process index queue items in the repository until you manually start an index agent.

The dmfulltext.ini file

The dmfulltext.ini file is created when the server is installed. It contains information used by Content Server to find the index agent plug-in.

Appendix A

Pre-installation Checklist

Use the checklist below to ensure that you have performed all required tasks before installing the full-text indexing software.

Full-text indexing checklist

Before installing a new server and repository or upgrading an existing repository, complete the following checklist, which contains a list of tasks that must be performed in order to prepare for implementing full-text indexing. (This checklist also appears in the *Content Server Installation Guide*.)

Table A-1. Checklist for Full-Text Indexing

Task	For More Information	Completed?
Determine which full-text indexing configuration to use	Chapter 2, Full-Text Indexing Deployment Models and Chapter 3, Planning considerations	
Determine the port numbers the index agent uses	Which ports to use for the index agent, page 47	
Determine the base port the index server uses	Which ports to use for the index server, page 48	

Task	For More Information	Completed?
Ensure that 4,000 ports above the base port number are unused and available for index server (for example, if the base port number is 3,000, port numbers from 3,000 to 7,000 must be available)	Which ports to use for the index server, page 48	
Determine the user account(s) to use for installing the index agent and index server	Index agent and index server installation account, page 50	
Ensure that the user accounts are set up	Network administrators	
Ensure that the disk space and memory requirements for the indexing software are met	Chapter 4, Preparing to Install Full-Text Indexing	
For a Content Server upgrade, decide whether to migrate the indexes before or after the server and repository are upgraded	Migrating the full-text indexing system, page 59	
Determine where the full-text indexes are to be created and stored	Disk space requirements for indexing and installation, page 46	
Ensure that any additional hosts required by the index agent and index server are configured	Chapter 4, Preparing to Install Full-Text Indexing	

Task	For More Information	Completed?
Mount the drive where the content files are located to make the content accessible to the index server(s)	Documentation from the hardware and operating system vendors; Deciding whether to share the drives where content files are located, page 52	
Determine for which languages grammatical normalization is required	Whether to use grammatical normalization , page 43	

Appendix B

Uninstalling the Index Agent and Index Server

This appendix describes how to uninstall the index agent and index server and how to delete an index. The appendix contains the following sections:

Order of uninstalling

You must use a particular order to uninstall Content Server, a repository, the index agent, and the index server.

To uninstall an index agent, the repository it servers must be running. To uninstall an index server, the repository must be shut down. If the index server is on the Content Server host, additional issues arise because of shared libraries in the software installations.

Uninstall the software components in this order:

1. Shut down and uninstall the index agent.
2. Shut down the repository.
3. Shut down and uninstall the index server.
4. Delete the repository, if required.
5. Uninstall the Content Server software, if required.
6. Uninstall the Index Agent Configuration Program, if required.

Deleting an index agent

Use these instructions to uninstall an index agent. .

To uninstall an index agent:

1. Log in to the index agent host as the user who installed the index agent.
2. Stop the index agent.

   Use the instructions in Starting and stopping the index agent, page 105 or in Documentum Administrator online Help, depending on which repository version the index agent is running against.
3. Start the Index Agent Configuration Program.
   - On Windows, click **Start→Programs→Documentum→Index Agent Configuration Program**.
   - On UNIX and Linux, navigate to $DOCUMENTUM_SHARED/IndexAgents and start the configuration program for your operating system:
     - On AIX, IndexAgent_Configuration_Program.aix
     - On Solaris, IndexAgent_Configuration_Program.bin
     - On HP-UX, IndexAgent_Configuration_Program.hp
     - On HP Itanium, IndexAgent_Configuration_ProgramHPIA64.bin
     - On Linux, IndexAgent_Configuration_Program.linux

   A Welcome dialog box is displayed.
4. Click **Next**.
5. Select **Delete index agent** and click **Next**.
6. Read the information and click **Next**.

   The index agent is deleted.
7. To run the configuration program again, check the check box.
8. Click **Next**.
9. If you checked the check box to run the configuration program again, skip to Step 5; otherwise, the program exits.

   The index agent software and configuration program are still on the host. To remove the software and configuration program, use the instructions in

Deleting the index agent configuration program

Use these instructions to delete the index agent configuration program from a host on which it is installed.

To delete the index agent configuration program:

1. Log in to the host as the user who installed the software.
2. If you are on Windows:
   a. Click **Start→Settings→Add/Remove Programs.**
   b. Select **Documentum Index Agent Configuration Program**.
   c. Click **Change/Remove**.
   The uninstaller starts.
3. If you are on UNIX or Linux, navigate to $DOCUMENTUM_SHARED/_uninst/IndexAgents and type `uninstall.bin`.
   The uninstaller starts.
4. Read the information screen and click **Next**.
   An information screen displays the location where the index agent software is installed.
5. Click **Next**.
   The software is uninstalled.
6. Click **Finish**.

Deleting an index server

Use these instructions to delete an index sever from the host.

To delete an index server:

1. Ensure that the repository served by the index server is stopped.
   Refer to the *Content Server Installation Guide* for information on stopping repositories.
2. Log in to the host where the index server is installed as the installation owner.
3. Use the instructions in Starting and stopping the index agent, page 105 or in Documentum Administrator online Help to stop the index server.
4. If you are on Windows:
   a. Click **Start→Settings→Control Panel→Add/Remove Programs.**

b. Select **Documentum Index Server**.

c. Click **Change/Remove**.

The uninstaller starts and a welcome dialog box is displayed.

5. If you are on UNIX or Linux, navigate to $DOCUMENTUM/_uninst/IndexServer and type `uninstall.bin`.

   The uninstaller starts and a welcome dialog box is displayed.

6. Click **Next**.

   An information dialog box is displayed with information about where the index server is installed.

7. Click **Next**.

   The software is uninstalled and a summary dialog box is displayed.

8. Click **Finish**.

Deleting a full-text Index

To delete a full-text index, run the uninstaller and delete the index agent and index server. A dialog box provides the option to delete the index as well.

Appendix C

Sample Output of ftintegrity Utility

The following report is the output of a failed completeness and accuracy run of the ftintegrity utility:

```
29 C% cobra ftintegrity.py -i ftintegrity.params.txt -m b
Mon Feb 07 15:48:37 2005 [PROGRESS]: Completeness Test Started
Mon Feb 07 15:48:38 2005 [PROGRESS]: Fetched all 11 docids from Documentum
Mon Feb 07 15:48:38 2005 [WARNING ]: Suspiciously low document count. Following
errors could be due to invalid result from the command below.
Mon Feb 07 15:48:38 2005 [WARNING ]: Please run it to validate correctness
Mon Feb 07 15:48:38 2005 [WARNING ]: C:\PROGRA~1\DOCUME~1\java\1449C1~1.2_0\bin\
java -classpath C:\PROGRA~1\DOCUME~1\INDEXA~1\INDEXA~1\webapps\INDEXA~1\WEB-INF\
lib\server-impl.jar;C:\PROGRA~1\DOCUME~1\dctm.jar;C:\Documentum\config com.docum
entum.server.impl.fulltext.ftintegrity.FTDumpIDs -m:0 -D:waasdfasdfasfasfasdfasf
 -U:wongw -P:groucho2
Mon Feb 07 15:48:38 2005 [INFO    ]: Get Documentum IDs: 0.988 s
Mon Feb 07 15:48:38 2005 [DEBUG   ]: Got 11 docids from Dctm
Mon Feb 07 15:48:38 2005 [INFO    ]: Get Number of Documents in Collection: 0.28
1 s
Mon Feb 07 15:48:38 2005 [INFO    ]: Number of documents in collection documentu
m: 1351
Mon Feb 07 15:48:39 2005 [PROGRESS]: Fetched all 1351 docids from FAST
Mon Feb 07 15:48:39 2005 [INFO    ]: Get FAST IDs: 0.780 s
Mon Feb 07 15:48:39 2005 [DEBUG   ]: Got 1351 docids from fast
Mon Feb 07 15:48:39 2005 [INFO    ]: Transform FAST ID file: 0.084 s
Mon Feb 07 15:48:39 2005 [INFO    ]: Transform Documentum ID file: 0.026 s
Mon Feb 07 15:48:39 2005 [INFO    ]: Compare IDs: 0.064 s
Mon Feb 07 15:48:39 2005 [PROGRESS]: Completeness Test Finished
Mon Feb 07 15:48:39 2005 [PROGRESS]: Accuracy Test Started
Mon Feb 07 15:48:39 2005 [DEBUG   ]: Getting Dctm Ids from res-comp-common.txt
Mon Feb 07 15:48:39 2005 [FATAL   ]: Failed to read any docids from Documentum.
The command below failed.
type res-comp-common.txt
Terminated.
```

The following report is the output of a successful completeness and accuracy run of the ftintegrity tool:

```
30 C% cobra ftintegrity.py -i ftintegrity.params.txt -m b
Mon Feb 07 15:50:29 2005 [PROGRESS]: Completeness Test Started
Mon Feb 07 15:50:35 2005 [PROGRESS]: Fetched all 1351 docids from Documentum
Mon Feb 07 15:50:35 2005 [INFO    ]: Get Documentum IDs: 5.223 s
Mon Feb 07 15:50:35 2005 [DEBUG   ]: Got 1351 docids from Dctm
Mon Feb 07 15:50:35 2005 [INFO    ]: Get Number of Documents in Collection: 0.21
9 s
```

```
Mon Feb 07 15:50:35 2005 [INFO    ]: Number of documents in collection documentu
m: 1351
Mon Feb 07 15:50:36 2005 [PROGRESS]: Fetched all 1351 docids from FAST
Mon Feb 07 15:50:36 2005 [INFO    ]: Get FAST IDs: 0.819 s
Mon Feb 07 15:50:36 2005 [DEBUG   ]: Got 1351 docids from fast
Mon Feb 07 15:50:36 2005 [INFO    ]: Transform FAST ID file: 0.114 s
Mon Feb 07 15:50:36 2005 [INFO    ]: Transform Documentum ID file: 0.099 s
Mon Feb 07 15:50:36 2005 [INFO    ]: Compare IDs: 0.136 s
Mon Feb 07 15:50:36 2005 [PROGRESS]: Completeness Test Finished
Mon Feb 07 15:50:36 2005 [PROGRESS]: Accuracy Test Started
Mon Feb 07 15:50:36 2005 [DEBUG   ]: Getting Dctm Ids from res-comp-common.txt
Mon Feb 07 15:50:36 2005 [PROGRESS]: Fetched all 1351 docids from Documentum
Mon Feb 07 15:50:36 2005 [PROGRESS]: Starting test of 0 documents
Mon Feb 07 15:50:36 2005 [PROGRESS]: Launching iftdql: C:\PROGRA~1\DOCUME~1\INDE
XA~1\FTINTE~1\iftdql C:\PROGRA~1\DOCUME~1\INDEXA~1\FTINTE~1\FASTQueryPlugin.dll

Mon Feb 07 15:51:03 2005 [PROGRESS]: Tested 100 of 1351 documents, 0 failures.
Estimated completion in 5.63 minutes
Mon Feb 07 15:51:16 2005 [PROGRESS]: Tested 200 of 1351 documents, 0 failures.
Estimated completion in 3.87 minutes
Mon Feb 07 15:51:28 2005 [PROGRESS]: Tested 300 of 1351 documents, 0 failures.
Estimated completion in 3.06 minutes
Mon Feb 07 15:51:43 2005 [PROGRESS]: Tested 400 of 1351 documents, 0 failures.
Estimated completion in 2.66 minutes
Mon Feb 07 15:51:57 2005 [PROGRESS]: Tested 500 of 1351 documents, 0 failures.
Estimated completion in 2.28 minutes
Mon Feb 07 15:52:07 2005 [PROGRESS]: Tested 600 of 1351 documents, 0 failures.
Estimated completion in 1.90 minutes
Mon Feb 07 15:52:17 2005 [PROGRESS]: Tested 700 of 1351 documents, 0 failures.
Estimated completion in 1.57 minutes
Mon Feb 07 15:52:29 2005 [PROGRESS]: Tested 800 of 1351 documents, 0 failures.
Estimated completion in 1.30 minutes
Mon Feb 07 15:52:42 2005 [PROGRESS]: Tested 900 of 1351 documents, 0 failures.
Estimated completion in 1.05 minutes
Mon Feb 07 15:52:54 2005 [PROGRESS]: Tested 1000 of 1351 documents, 0 failures.
Estimated completion in 0.81 minutes
Mon Feb 07 15:53:06 2005 [PROGRESS]: Tested 1100 of 1351 documents, 0 failures.
Estimated completion in 0.57 minutes
Mon Feb 07 15:53:17 2005 [PROGRESS]: Tested 1200 of 1351 documents, 0 failures.
Estimated completion in 0.34 minutes
Mon Feb 07 15:53:29 2005 [PROGRESS]: Tested 1300 of 1351 documents, 0 failures.
Estimated completion in 0.11 minutes
Mon Feb 07 15:53:34 2005 [INFO    ]: Total documents:  1351
Mon Feb 07 15:53:35 2005 [INFO    ]: Documents tested: 1351
Mon Feb 07 15:53:35 2005 [INFO    ]: Failures:         0
Mon Feb 07 15:53:35 2005 [INFO    ]: Statistical Information:
Mon Feb 07 15:53:35 2005 [INFO    ]:        Min        Max        Avg     StdDev
  N Name
Mon Feb 07 15:53:35 2005 [INFO    ]:       0.01       6.80       0.03       0.19
 1351 Dump Attributes
Mon Feb 07 15:53:35 2005 [INFO    ]:       0.00       0.00       0.00       0.00
 1351 Build Query
Mon Feb 07 15:53:35 2005 [INFO    ]:       0.01       1.16       0.10       0.10
 1351 Execute Query
Mon Feb 07 15:53:35 2005 [INFO    ]:       0.02       7.04       0.13       0.22
 1351 Test One Document
```

Appendix D

Supported and Unsupported Formats for Full-Text Indexing

The formats listed in Table D–1, page 145are supported for full-text indexing. Formats not listed in Table D–1, page 145 are not supported.

Table D-1. Supported document formats

Document format	Version
Access	Versions through 2.0
Adobe Acrobat (PDF)	Versions 2.1, 3.0 – 7.0; Note that scanned PDFs cannot be searched unless the OCR equipment generated searchable PDFs.
Adobe FrameMaker (MIF)	Version 6.0
Adobe FrameMaker graphics (FMV)	Vector/raster through 5.0
Adobe Illustrator	Versions through 7.0, 9.0
Adobe Photoshop (PSD)	Version 4.0
Ami Draw (SDW)	Ami Draw
ANSI Text	7 & 8 bit
ASCII Text	7 & 8 bit
AutoCAD Drawing	Versions 2.5 - 2.6, 9.0 - 14.0, 2000i and 2002
AutoCAD Interchange and Native Drawing formats	DXF and DWG
AutoShade Rendering (RND)	Version 2.0
Binary Group 3 Fax	All versions

Document format	Version
Bitmap (BMP, RLE, ICO, CUR, OS/2 DIB & WARP)	All versions
CALS Raster (GP4)	Type I and Type II
Computer Graphics Metafile (CGM)	ANSI, CALS NIST version 3.0
Corel Clipart format (CMX)	Versions 5 through 6
Corel Draw (CDR with TIFF header)	Versions 2.x – 9.x
Corel Draw (CDR)	Versions 3.x – 8.x
Corel/Novell Presentations	Versions through 11.0
DataEase	Version 4.x
dBASE	Versions through 5.0
dBXL	Version 1.3
DEC WPS Plus (DX)	Versions through 4.0
DEC WPS Plus (WPL)	Versions through 4.1
DisplayWrite 2 & 3 (TXT)	All versions
DisplayWrite 4 & 5	Versions through Release 2.0
EBCDIC	
Enable	Versions 3.0, 4.0 and 4.5
Enable	Versions 3.0, 4.0 and 4.5
Encapsulated PostScript (EPS)	TIFF header only
Executable (EXE, DLL)	
Executable (Windows) NT	
First Choice	through 3.0
First Choice	Versions through 3.0
First Choice	Versions through 3.0
FoxBase	Version 2.1
Framework	3.0
Framework	Version 3.0
Framework	Version 3.0
Freelance (Windows)	Versions through Millennium 9.6
Freelance for OS/2	Versions through 2.0
GEM Paint (IMG)	No specific version

Document format	Version
Graphics Environment Mgr (GEM)	Bitmap & vector
Graphics Interchange Format (GIF)	No specific version
GZIP	All versions
Hangul	Version 97
Harvard Graphics (Windows)	Windows versions
Harvard Graphics for DOS	Versions 2.x & 3.x
Hewlett Packard Graphics Language (HPGL)	Version 2
HTML	through 3.0 (some limitations)
IBM FFT	All versions
IBM Graphics Data Format (GDF)	Version 1.0
IBM Picture Interchange Format (PIF)	Version 1.0
IBM Revisable Form Text	All versions
IBM Writing Assistant	1.01
Initial Graphics Exchange Spec (IGES)	Version 5.1
JFIF (JPEG not in TIFF format)	All versions
JPEG (including EXIF)	All versions
JustSystems Ichitaro	Versions 5.0, 6.0, 8.0 – 12.0
JustWrite	Versions through 3.0
Kodak Flash Pix (FPX)	All versions
Kodak Photo CD (PCD)	Version 1.0
Legacy	Versions through 1.1
Lotus 1-2-3 (DOS & Windows)	Versions through 5.0
Lotus 1-2-3 (OS/2)	Versions through 2.0
Lotus 1-2-3 Charts (DOS & Windows)	Versions through 5.0
Lotus 1-2-3 for SmartSuite	Versions 97 – Millennium 9.6
Lotus AMI/AMI Professional	Versions through 3.1
Lotus Manuscript	Version 2.0
Lotus PIC	All versions
Lotus Snapshot	All versions
Lotus Symphony	Versions 1.0,1.1 and 2.0

Document format	Version
Lotus Word Pro	Versions 96 through Millennium Edition 9.6, text only
LZA Self Extracting Compress	All versions
LZH Compress	All versions
Macintosh PICT1 & PICT2	Bitmap only
MacPaint (PNTG)	No specific version
MacWrite II	Version 1.1
MASS11	Versions through 8.0
Micrografx Designer (DRW)	Versions through 3.1
Micrografx Designer (DSF)	Windows 95, version 6.0
Micrografx Draw (DRW)	Versions through 4.0
Microsoft Binder	Versions 7.0-97 (only on Windows)
Microsoft Excel (Mac)	Versions 3.0 – 4.0, 98, 2001
Microsoft Excel (Windows)	Versions 2.2 through 2003
Microsoft Excel Charts	Versions 2.x - 7.0
Microsoft Multiplan	Version 4.0
Microsoft Outlook Folder (PST)	Versions 97, 98, 2002, and 2002
Microsoft Outlook Message (MSG)	All versions. The body of the message is indexed and attachments to messages are indexed.
Microsoft PowerPoint (Mac)	Versions 4.0 through 2001
Microsoft PowerPoint (Windows)	Versions 3.0 through 2003
Microsoft Project	Versions 98 - 2002 (text only)
Microsoft Rich Text Format (RTF)	All versions
Microsoft Word	Versions through 6.0
Microsoft Word	Versions through 2003
Microsoft Word (Mac)	Versions 3.0 – 4.0, 98, 2001
Microsoft WordPad	All versions
Microsoft Works	Versions through 2.0
Microsoft Works	Versions through 4.0
Microsoft Works (DOS)	Versions through 2.0

Document format	Version
Microsoft Works (DOS)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Microsoft Works (Windows)	Versions through 4.0
Microsoft Works (Windows)	Versions through 4.0
Microsoft Write	Versions through 3.0
MIME Text Mail	
Mosaic Twin	Version 2.5
MultiMate	Versions through 4.0
Navy DIF	All versions
Nota Bene	Version 3.0
Novell Perfect Works	Version 2.0
Novell Perfect Works	Version 2.0
Novell Perfect Works (Draw)	Version 2.0
Novell WordPerfect	Versions through 6.1
Novell WordPerfect	Versions 1.02 through 3.0
Novell/Corel WordPerfect	Versions through 11.0
Office Writer	Versions 4.0 - 6.0
OS/2 PM Metafile (MET)	Version 3.0
Paint Shop Pro 6 (PSP)	Windows only, versions 5.0 – 6.0
Paradox (DOS)	Versions through 4.0
Paradox (Windows)	Versions through 1.0
PC Paintbrush (PCX and DCX)	All versions
PC-File Letter	Versions through 5.0
PC-File+ Letter	Versions through 3.0
Personal R:BASE	Version 1.0
PFS: Professional Plan	Version 1.0
PFS: Write	Versions A, B and C
Portable Bitmap (PBM)	All versions
Portable Graymap (PGM)	No specific version

Document format	Version
Portable Network Graphics (PNG)	Version 1.0
Portable Pixmap (PPM)	No specific version
Postscript (PS)	Level II
Professional Write	Versions through 2.1
Professional Write Plus	Version 1.0
Progressive JPEG.	No specific version
Q&A	Versions through 2.0
Q&A	Version 2.0
Q&A Write	Version 3.0
Quattro Pro (DOS)	Versions through 5.0
Quattro Pro (Windows)	Versions through 11.0
R:BASE 5000	Versions through 3.1
R:BASE System V	Version 1.0
Reflex	Version 2.0
Samna Word	Versions through Samna Word IV+
SmartWare II	Version 1.02
SmartWare II	Version 1.02
SmartWare II	Version 1.02
Sprint	Versions through 1.0
Star Office/Open Office Calc	Star Office Versions 5.2, 6.x, and 7.x; Open Office version 1.1 (text only)
Star Office/Open Office Draw	Star Office 5.2, 6.x, and 7.x; Open Office version 1.1 (text only)
Star Office/Open Office Writer	Star Office Versions 5.2, 6.x, and 7.x; Open Office version 1.1 (text only)
StarOffice / Open Office Impress	StarOffice 5.2, 6.x, and 7.x; Open Office 1.1 (text only)
Sun Raster (SRS)	No specific version
SuperCalc 5	Version 4.0
Text Mail (MIME)	
TIFF	Versions through 6
TIFF CCITT Group 3 & 4	Versions through 6

Document format	Version
Total Word	Version 1.2
Truevision TGA (TARGA)	Version 2
Unicode Text	All versions
UNIX Compress	
UNIX TAR	
UUEncode	
vCard	Version 2.1
Visio	Versions 5, 2000 and 2002
Visio (preview)	Version 4
Volkswriter 3 & 4	Versions through 1.0
VP Planner 3D	Version 1.0
Wang PC (IWP)	Versions through 2.6
WBMP	No specific version
Windows Enhanced Metafile (EMF)	No specific version
Windows Metafile (WMF)	No specific version
WML	Version 5.2
WordMARC	Versions through Composer Plus
WordPerfect Graphics (WPG & WPG2)	Versions through 2.0, 7 and 10
WordStar	Versions through 7.0
WordStar	Version 1.0
WordStar 2000	Versions through 3.0
XML	All versions
X-Windows Bitmap (XBM)	x10 compatible
X-Windows Dump (XWD)	x10 compatible
X-Windows Pixmap (XPM)	x10 compatible
XyWrite	Versions through III Plus
ZIP	PKWARE versions through 2.04g. Files inside a ZIP file are *not* indexed.

The formats listed in Table D–2, page 152are not supported for full-text indexing. Only those formats listed in Table D–1, page 145 are supported.

Table D-2. Unsupported document formats

Document format	Version
Applix Graphics	4.3, 4.4
Applix Presents	4.3, 4.4
Applix Spreadsheets	4.2, 4.3, 4.4
Applix Words	4.2, 4.3, 4.4
Folio Flat File	3.1
Fujitsu Oasys	7.0
Microsoft Outlook Express eml format	No specific version
Microsoft Windows Animated Cursor	No specific version
Microsoft Windows Cursor/Icon	No specific version
SGI RGB	No specific version

Appendix E

Supported Languages for Full-Text Indexing

This appendix lists the languages supported for full-text indexing.

The following languages are supported for full-text indexing:

Table E-1. Supported languages

Language	Code	Language	Code
Afrikaans	af	Italian	it
Albanian	sq	Japanese	na
Arabic	ar	Kazahk	kk
Armenian	hy	Kirghiz	ky
Azeri	az	Korean	ko
Bangla	bn	Kurdish	ku
Basque	eu	Latin	la
Bosnian	bs	Latvian	lv
Breton	br	Letzeburgesch	lb
Bulgarian	bg	Lithuanian	lt
Byelorussian	by	Macedonian	mk
Catalan	ca	Malay	ms
Chinese_simplified	zh_cn	Maltese	mt
Chinese_traditional	zh_tw	Maori	mi
Croatian	hr	Mongolian	mn
Czech	cs	Norwegian_ Bokmaal	nb

Language	Code	Language	Code
Danish	da	Norwegian_ Nynorsk	nn
Dutch	nl	Polish	pl
English	en	Portuguese	pt
Esperanto	eo	Rhaeto_Romance (Romansch)	rm
Estonian	et	Romanian	ro
Faeroese	fo	Russian	ru
Farsi	fa	Sami_Northern	se
Filipino (Tagalog)	tl	Serbian	sr
Finnish	fi	Slovak	sk
French	fr	Slovenian	sl
Frisian	fy	Spanish	es
Galician	gl	Swahili	sw
Georgian	ka	Swedish	sv
German	de	Tamil	ta
Greek	el	Thai	th
Greenlandic	kl	Turkish	tr
Hausa	ha	Ukrainian	uk
Hebrew	he	Urdu	ur
Hindi	hi	Uzbek	uz
Hungarian	hu	Vietnamese	vi
Icelandic	is	Welsh	cy
Indonesian	id	Yiddish	yi
Irish_Gaelic	ga	Zulu	zu

Index

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X