



## OpenText™ Documentum™ xPlore

### **Administration and Development Guide**

This guide describes administration, configuration, and customization of Documentum xPlore.

EDCSRC220100-AGD-EN-02

---

## **OpenText™ Documentum™ xPlore Administration and Development Guide**

EDCSRC220100-AGD-EN-02

Rev.: 2023-Oct-05

This documentation has been created for OpenText™ Documentum™ xPlore CE 22.1.

It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

### **Open Text Corporation**

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111

Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440

Fax: +1-519-888-0677

Support: <https://support.opentext.com>

For more information, visit <https://www.opentext.com>

### **Copyright © 2023 Open Text.**

One or more patents may cover this product(s). For more information, please visit <https://www.opentext.com/patents>.

### **Disclaimer**

#### **No Warranties and Limitation of Liability**

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

---

# Table of Contents

<b>1</b>	<b>Introduction to xPlore .....</b>	<b>13</b>
1.1	Features .....	13
1.2	Limitations .....	15
1.3	xPlore compared to FAST .....	18
1.4	Architectural overview .....	20
1.5	Physical architecture .....	20
1.5.1	xPlore disk areas .....	21
1.5.2	xPlore instances .....	22
1.5.3	xDB libraries and Lucene index .....	23
1.5.4	Indexes .....	25
1.6	Logical architecture .....	25
1.6.1	Documentum domains and categories .....	27
1.6.2	Mapping of domains to xDB .....	29
1.7	How Documentum Server documents are indexed .....	31
1.8	How Documentum Server documents are queried .....	33
<b>2</b>	<b>Managing the System .....</b>	<b>35</b>
2.1	Opening xPlore Administrator .....	35
2.2	Managing user access .....	35
2.2.1	Creating users, groups and deployments .....	36
2.2.2	Changing and resetting passwords .....	36
2.3	Starting and stopping the system .....	37
2.4	Viewing and configuring global operations (all instances) .....	37
2.5	Configuring an instance .....	38
2.6	Changing the host name and URL .....	38
2.7	Replacing a failed instance with a spare .....	40
2.8	Replacing a failed primary instance .....	41
2.9	Changing a failed instance into a spare .....	43
2.10	Starting and stopping the watchdog service .....	44
2.11	Configuring the watchdog service .....	44
2.12	Configuring disk space monitoring .....	46
2.13	Configuring xDB log monitoring .....	47
2.14	Configuring system metrics .....	48
2.15	Configuring the audit record .....	48
2.15.1	Anonymizing Search Users .....	49
2.16	Troubleshooting system problems .....	50
2.17	Modifying indexserverconfig.xml .....	54
2.17.1	Customizations in indexserverconfig.xml .....	56
2.18	Tasks performed outside xPlore administrator .....	56
2.19	Administration APIs .....	58

2.19.1	Open an admin connection .....	59
2.19.2	Call an admin API .....	59
2.19.3	Configuration APIs .....	59
<b>3</b>	<b>Managing Security .....</b>	<b>63</b>
3.1	About security .....	63
3.2	Changing search results security .....	63
3.3	Manually updating security .....	64
3.4	Changing the xDB administrator password .....	66
3.5	Configuring the security cache .....	67
3.6	Troubleshooting security .....	68
<b>4</b>	<b>Managing the Documentum Index Agent .....</b>	<b>73</b>
4.1	About the Documentum index agent .....	73
4.1.1	Documentum attributes that control indexing .....	74
4.2	Starting the index agent .....	76
4.3	Silent index agent startup .....	77
4.4	Manipulating an Index Agent with DM_TICKET .....	80
4.5	Setting up index agents for ACLs and groups .....	80
4.6	Configuring the index agent after installation .....	81
4.6.1	Overview of Index Agent filters .....	82
4.6.2	Configuring Index Agent filters .....	85
4.6.3	Monitor a running Index Agent .....	86
4.6.4	Sharing content storage .....	88
4.6.5	Mapping Server storage areas to collections .....	90
4.7	Migrating documents .....	92
4.7.1	Migrating content (reindexing) .....	92
4.7.2	Migrating documents by object type .....	92
4.7.3	Migrating a limited set of documents .....	93
4.7.4	Moving collection documents without reindexing .....	93
4.8	Using ftintegrity .....	94
4.8.1	ftintegrity output .....	96
4.8.2	ftintegrity result files .....	97
4.8.3	Running the state of index job .....	98
4.8.4	state of index and ftintegrity arguments .....	98
4.9	Indexing documents in normal mode .....	103
4.10	Resubmitting documents for indexing .....	103
4.11	Indexing auto retry .....	104
4.11.1	Disabling indexing auto retry .....	105
4.11.2	Enable indexing auto retry for documents indexed with warnings .....	106
4.12	Removing entries from the index .....	106
4.13	Indexing metadata only .....	107
4.14	Making types non-indexable .....	107

4.15	Making metadata non-searchable .....	107
4.16	Injecting data and supporting joins .....	108
4.17	Reindexing after removing a Documentum attribute .....	111
4.18	Troubleshooting the index agent .....	111
4.18.1	Cleaning up the index queue .....	116
4.18.2	Index agent startup issues .....	117
4.18.3	Content query returned no data .....	118
<b>5</b>	<b>Document Processing (CPS) .....</b>	<b>119</b>
5.1	About CPS .....	119
5.2	Adding a remote CPS instance .....	120
5.3	Removing a CPS instance .....	121
5.4	Configuring CPS dedicated to indexing or search .....	122
5.5	Administering CPS .....	122
5.6	Modifying CPS configuration file .....	123
5.7	Maximum document and text size .....	124
5.8	Configuring languages and encoding .....	126
5.9	Indexable formats .....	131
5.10	Lemmatization .....	131
5.10.1	About lemmatization .....	131
5.10.2	Configuring indexing lemmatization .....	133
5.10.3	Lemmatizing specific types or attributes .....	133
5.10.4	Troubleshooting lemmatization .....	135
5.10.5	Saving lemmatization tokens .....	136
5.10.6	Defining disambiguation rules .....	137
5.11	Handling special characters .....	138
5.12	Configuring stop words .....	141
5.13	Searching Chinese, Japanese, or Korean Terms .....	143
5.14	Troubleshooting content processing .....	143
5.14.1	CPS troubleshooting methods .....	143
5.14.2	CPS startup and connection errors .....	145
5.14.3	Adding CPS daemons for ingestion or query processing .....	146
5.14.4	Running out of space .....	147
5.14.5	CPS file processing errors .....	147
5.14.6	Troubleshooting slow ingestion and timeouts .....	149
5.15	Adding dictionaries to CPS .....	152
5.16	Custom content processing .....	155
5.16.1	About custom content processing .....	155
5.16.2	Text extraction .....	157
5.16.3	Troubleshooting custom text extraction .....	160
5.16.4	Annotation .....	160
5.16.5	Apache UIMA example .....	165

5.16.6	Custom content processing errors .....	169
<b>6</b>	<b>Indexing .....</b>	<b>171</b>
6.1	About indexing .....	171
6.2	Configuring text extraction .....	171
6.3	Configuring an index .....	173
6.3.1	Subpaths .....	176
6.3.2	Sort support .....	178
6.3.3	XML attribute support .....	180
6.4	Creating custom indexes .....	181
6.5	Managing indexing in xPlore administrator .....	181
6.6	Troubleshooting indexing .....	182
6.7	Running the standalone data consistency checker .....	185
6.8	Indexing APIs .....	187
6.8.1	Route a document to a collection .....	187
6.8.1.1	Creating a custom routing class .....	187
6.8.1.2	SimpleCollectionRouting example .....	188
<b>7</b>	<b>Index Data: Domains, Categories, and Collections .....</b>	<b>191</b>
7.1	Domain and collection menu actions .....	191
7.2	Managing domains .....	192
7.3	Delete a corrupted domain .....	193
7.4	Configuring categories .....	194
7.5	Managing collections .....	195
7.5.1	About collections .....	195
7.5.2	Planning collections for scalability .....	196
7.5.3	Uses of subcollections .....	197
7.5.4	Adding or deleting a collection .....	197
7.5.5	Changing collection properties .....	198
7.5.6	Routing documents to a specific collection .....	199
7.5.7	Routing to specific collections using routing classes .....	199
7.5.8	Attaching and detaching a collection .....	202
7.5.9	Moving a collection .....	202
7.5.10	Moving a collection to a new drive .....	203
7.5.11	Creating a storage location .....	204
7.5.12	Monitoring merges of index data .....	204
7.5.13	Querying a collection .....	204
7.6	Rebuilding indexes .....	205
7.6.1	Rebuilding indexes through data refeed .....	205
7.6.2	Configuring online index rebuilds .....	206
7.6.3	Performing an online index rebuild .....	206
7.6.4	Deleting a collection and recreating indexes .....	207
7.7	Checking xDB statistics .....	208

7.8	Troubleshooting data management .....	209
7.9	xDB repair commands .....	210
<b>8</b>	<b>Backup and Restore .....</b>	<b>213</b>
8.1	About backup .....	213
8.2	About restore .....	216
8.3	Performing a native xDB backup in xPlore Administrator .....	217
8.4	Performing an offline restore .....	218
8.5	Performing a remote restore .....	219
8.6	Performing a file- or volume-based (snapshot) backup and restore ...	220
8.7	Troubleshooting backup and restore .....	220
8.8	Handling data corruption .....	222
8.8.1	Detecting data corruption .....	222
8.8.2	Handling a corrupt domain .....	222
8.8.3	Repairing a corrupted index .....	222
8.8.4	Cleaning and rebuilding the index .....	223
8.8.5	Dead objects .....	224
8.8.6	Recovering from a system crash .....	226
<b>9</b>	<b>Automated Utilities (CLI) .....</b>	<b>227</b>
9.1	CLI properties and environment .....	227
9.2	Using the CLI .....	227
9.3	CLI batch file .....	228
9.4	Scripted backup .....	229
9.5	Scripted federation restore .....	230
9.6	Scripted domain restore .....	230
9.7	Scripted collection restore .....	231
9.8	Force detach and attach CLIs .....	232
9.9	Orphaned segments CLIs .....	233
9.10	Domain mode CLIs .....	234
9.11	Collection and domain state CLIs .....	234
9.12	Activate spare instance CLI .....	235
9.13	Detecting the version of an instance .....	235
9.14	Cleaning up after failed index rebuild .....	235
9.15	Final merge CLIs .....	236
9.16	Collecting diagnostic information .....	236
<b>10</b>	<b>Search .....</b>	<b>239</b>
10.1	About searching .....	239
10.1.1	Query operators .....	240
10.2	Administering search .....	241
10.2.1	Configuring query warmup .....	242
10.2.2	Configuring scoring and freshness .....	246

10.2.3	Supporting search in XML documents .....	247
10.2.4	Adding a thesaurus .....	252
10.2.5	Configuring query lemmatization .....	257
10.2.6	Configuring search on compound terms .....	257
10.2.7	Using query summaries .....	258
10.2.8	Configuring query summaries .....	259
10.2.8.1	Highlighting query summaries .....	261
10.2.8.2	Highlighting query metadata summaries .....	262
10.2.8.3	Configuring summary security .....	263
10.2.8.4	Configuring parallel summary calculation .....	263
10.2.9	Configuring fuzzy search .....	264
10.2.10	Configuring index type checking .....	266
10.3	Configuring a wildcard search .....	268
10.3.1	Configuring a leading-wildcard subpath .....	269
10.3.2	Limiting wildcards and common terms in search results .....	270
10.3.3	Supporting wildcard searches in DFC .....	271
10.3.4	Supporting wildcard searches in DQL .....	272
10.4	Configuring Documentum search .....	272
10.4.1	Limiting query fetch time .....	272
10.4.2	Query plugin configuration (dm_ftengine_config) .....	273
10.4.3	Making types and attributes searchable .....	274
10.4.4	Running folder descend queries .....	275
10.4.5	DQL, DFC, and DFS queries .....	276
10.4.6	Documentum Server and DFC client search differences .....	278
10.4.7	DQL Processing .....	278
10.4.8	Tracing Documentum queries .....	280
10.5	Supporting subscriptions to queries .....	281
10.5.1	About query subscriptions .....	281
10.5.2	Installing the query subscription DAR .....	285
10.5.3	Testing query subscriptions .....	286
10.5.4	Subscription reports .....	287
10.5.5	Subscription logging .....	288
10.5.6	dm_ftquery_subscription .....	289
10.5.7	dm_qbs_relation object .....	292
10.5.8	Query subscription jobs .....	293
10.5.9	Query subscription workflows .....	295
10.5.10	IQuerySubscriptionSBO .....	296
10.5.11	IQuerySubscriptionTBO .....	298
10.5.12	QuerySubscriptionAdminTool .....	299
10.6	Troubleshooting search .....	301
10.6.1	Auditing queries .....	302
10.6.2	Search is not available .....	304

10.6.3	Troubleshooting slow queries .....	305
10.6.4	Unexpected search results .....	308
10.6.5	Debugging queries .....	311
10.6.6	About Testing Text Extraction .....	315
10.6.7	Testing Text Extraction .....	316
10.6.8	Rerun a Search Query from an Audit Report .....	316
10.7	Routing a query to a specific collection .....	317
10.8	Debugging queries .....	318
10.9	Building a query with the DFC search service .....	319
10.10	Building a query with the DFS search service .....	319
10.11	Building a DFC XQuery .....	321
10.12	Building a query using xPlore APIs .....	323
10.13	Adding context to a query .....	326
10.14	Using parallel queries .....	327
10.15	Adding custom access to a thesaurus .....	328
10.16	Performing proximity searches .....	331
<b>11</b>	<b>Facets .....</b>	<b>333</b>
11.1	About Facets .....	333
11.2	Configuring facets in xPlore .....	334
11.3	Creating a DFC facet definition .....	336
11.4	Facet datatypes .....	336
11.5	Creating a DFS facet definition .....	339
11.6	Defining a facet handler .....	341
11.7	Sample DFC facet definition and retrieval .....	342
11.8	Tuning facets .....	344
11.9	Logging facets .....	345
11.10	Troubleshooting facets .....	345
<b>12</b>	<b>Using reports .....</b>	<b>347</b>
12.1	About reports .....	347
12.2	Types of reports .....	347
12.3	Document processing (CPS) reports .....	350
12.4	Indexing reports .....	350
12.5	Search reports .....	350
12.6	Editing a report .....	351
12.7	Report syntax .....	352
12.8	Sample edited report .....	354
12.9	Troubleshooting reports .....	356
<b>13</b>	<b>Logging .....</b>	<b>357</b>
13.1	Configuring logging .....	357
13.2	CPS logging .....	359

<b>14</b>	<b>Setting up a Customization Environment .....</b>	<b>361</b>
14.1	Setting up the xPlore SDK .....	361
14.2	Customization points .....	361
14.3	Adding custom classes .....	364
14.4	Enabling logging in a client application .....	364
14.5	Handling a NoClassDefFoundError exception .....	365
<b>15</b>	<b>Performance and Disk Space .....</b>	<b>367</b>
15.1	Planning for performance .....	367
15.2	Disk space and storage .....	369
15.3	System sizing for performance .....	372
15.4	Memory consumption .....	373
15.5	Measuring performance .....	374
15.6	Tuning the system .....	375
15.7	Tuning Lucene internal merges .....	376
15.8	Managing sub-index merges .....	377
15.8.1	Manually starting and stopping final merges .....	379
15.8.2	Managing final merges through interval-based or Cron-style scheduling .....	380
15.8.3	Final merging priorities and prioritization rules .....	381
15.8.4	mergethruschedule .....	383
15.8.5	Setting final merge blackout periods .....	385
15.8.6	How threshold-based scheduling works .....	386
15.9	Documentum index agent performance .....	387
15.10	Indexing performance .....	388
15.11	Search performance .....	389
15.11.1	About search performance .....	390
15.11.2	Tuning CPS and xDB for search .....	391
15.11.3	Creating a CPS daemon dedicated to search .....	393
15.11.4	Improving search performance with time-based collections .....	394
15.12	Throttling indexing and searching .....	394
<b>A</b>	<b>Index Agent, CPS, Indexing, and Search Parameters .....</b>	<b>397</b>
A.1	dm_ftengine_config .....	397
A.2	Index agent configuration parameters .....	399
A.3	Document processing and indexing service configuration parameters	402
A.4	Search service configuration parameters .....	407
A.5	API Reference .....	409
<b>B</b>	<b>The dftxml Category .....</b>	<b>413</b>
B.1	Extensible Documentum DTD .....	413
B.2	Supporting XML namespaces .....	416

<b>C</b>	<b>XQuery and VQL Reference .....</b>	<b>419</b>
C.1	Tracking XQueries .....	419
C.2	VQL and XQuery Syntax Equivalents .....	420
<b>D</b>	<b>xPlore Glossary .....</b>	<b>421</b>



# Chapter 1

## Introduction to xPlore

This guide describes administration, configuration, and customization of Documentum xPlore. This guide contains information for xPlore administrators who configure xPlore and Java developers who customize xPlore:

- Configuration is defined for support purposes as changing an XML file or an administration setting in the UI.
- Customization is defined for support purposes as using xPlore APIs to customize indexing and search. The xPlore SDK is a separate download that supports customization.

You must be familiar with the installation guide, which describes the initial configuration of the xPlore environment. You must also be familiar with Documentum Server administration.

### 1.1 Features

Documentum xPlore is a multi-instance, scalable, high-performance, full-text index server that can be configured for high availability and disaster recovery.

The xPlore architecture is designed with the following principles:

- Uses standards as much as possible, like XQuery.
- Uses open source tools and libraries, like Lucene.
- Supports virtualization, with accompanying lower total cost of ownership.
- Enterprise readiness: High availability, backup and restore, analytics, performance tuning, reports, diagnostics and troubleshooting, administration GUI, and configuration and customization points.
- Integration readiness: Fully supports XML namespaces—Indexing, storing, and querying XML documents with namespaces, configuring index paths that contain namespaces, and executing XQueries with namespace mappings.

#### Indexing features

Collection topography: xPlore supports creating collections online, and collections can span multiple file systems.

Transactional updates and purges: xPlore supports transactional updates and purges of indexes.

Multithreaded insertion into indexes: xPlore ingestion through multiple threads supports vertical scaling on the same host.

Dynamic allocation and deallocation of capacity: for periods of high ingestion, you can add a CPS instance deployed on a higher performance machine. You can also move a collection to another node for better performance.

Temporary high query load: For high query load, like a legal investigation, add an xPlore instance for the search service and bind collections to it in read-only mode.

Growing ingestion or query load: If your ingestion or query load increases due to growing business, you can add instances as needed.

Extensible indexing pipeline using the open-source UIMA framework.

Configurable stop words and special characters.

## Search features

Case sensitivity: xPlore queries are lower-cased (rendered case-insensitive).

Full-text queries: To query metadata, set up a specific index on the metadata.

Faceted search: Facets in xPlore are computed over the entire result set or over a configurable number of results.

Security evaluation: When a user performs a search, permissions are evaluated for each result. Security can be evaluated in the xPlore full-text engine before results are returned to Documentum Server, resulting in faster query results. This feature is turned on by default and can be configured or turned off.

Native XQuery syntax: The xPlore full-text engine supports XQuery syntax.

Thesaurus search to expand query terms.

Fuzzy search finds misspelled words or letter reversals.

Boost specific metadata in search results.

Extensive testing and validation of search on supported languages.

## Administration features

Multiple instance configuration and management.

Reports on ingestion metrics and errors, search performance and errors, and user activity.

Collections management: Creating, configuring, deleting, binding, routing, rebuilding, querying.

Command-line interface for automating data management (such as final merge), backup, and restore.

## 1.2 Limitations

### ACLs and aspects are not searchable by default

ACLs and aspects are not searchable by default, to protect security. You can reverse the default by editing indexserverconfig.xml. Set full-text-search to true in the subpath definition for acl\_name and r\_aspect\_name and then rebuild indexes of your data collections.

### Ingestion of many large files can cause failures

When CPS processes two or more large files at the same time, the CPS log file reports one of the following errors (cps\_daemon.log):

```
ERROR [Daemon-Core-(3400)] Exception happened, ACCESS_VIOLATION,
Attempt to read data at address 1 at (connection-handler-2)
...
FATAL [DAEMON-LP_RLP-(3440)] Not enough memory to process linguistic requests.
Error message: bad allocation
```

The workaround is to use xPlore administrator. Select an instance and click **Configuration**. Change the following to smaller values:

- Batch size: Decrease batch size to decrease the number of documents in a failed batch. All documents in a batch fail to be indexed if one document fails.
- Max text threshold
- Thread pool size

### Only metadata is indexed if a format is not supported

CPS cannot process certain formats as documents or email attachments, and PDF input fields. Only metadata of unsupported contents is indexed during ingestion. For a full list of supported formats, see *Oracle Outside In documentation*. To see format processing errors, use the xPlore administrator report, *Document Processing Error Detail* and choose *File format unsupported*. To test whether a format is supported, try uploading it in xPlore administrator. If no error code is reported in the Document processing error report, the format has been successfully indexed.

### Missing or Extra Spaces in PDF Files

The original design of PDF focused on accurate and consistent display across various platforms and devices. This brings the lack of structure within PDF files. The text on a page of PDF can be characterized as a sequence of glyphs drawn at x/y locations on the page canvas without any regard for structure. As a result, some phrases may not contain spaces between words while some words may contain extra spaces between letters. This third-party issue makes content extraction from PDF files problematic, and thus impacts the search result. For example, when the word *volleyball* is extracted as two words *volley ball*, a query with the keyword *volleyball* may miss some documents in the result set.

## CPS daemon must restart after fatal ingestion error

CPS daemon automatically restarts when ingestion encounters a fatal error. The CPS log indicates that the connection has been reset:

```
2009-02-10 12:19:55,425 INFO [DAEMON-CORE- (-1343566944)]  
Daemon is shutdown forcefully.  
2009-02-10 12:19:55,512 ERROR [MANAGER-CPSTransporter- (CPSWorkerThread-6)]  
Failed to receive the response XML.  
java.net.SocketException: Connection reset
```

## Batch failure

Indexing requests are processed in batches. When one request in a batch fails when the index is written to xDB, the entire batch fails.

## Collection names cannot be duplicated

The name of an adopted collection cannot be reused, because the name still exists in the domain.

An adopted collection is a collection that has been moved to a parent collection. The adopted collection becomes a subcollection. This kind of collection is created to boost ingestion rate, and later adopted for better search performance.

## Lemmatization

- xPlore supports lemmatization, but you cannot configure the parts of speech that are lemmatized.
- The part of speech for a word can be misidentified when there is not enough context. *Workaround:* Enable alternative lemmatization if you have disabled it (see “Configuring indexing lemmatization” on page 133).
- Punctuation at the end of the sentence is included in the lemmatization of the last word. For example, a phrase *Mary likes swimming and dancing* is lemmatized differently depending on whether there is a period at the end. Without the period, *dancing* is identified as a verb with the lemma *dance*. With the period, it is identified as a noun with the lemma *dancing*. A search for the verb *dance* does not find the document when the word is at the end of the sentence. The likelihood of errors in Part-Of-Speech (POS) tagging increases with sentence length.  
*Workaround:* Enable alternate lemmatization.

## Multilingual support

By default, documents with multiple languages are indexed by only one language. In the latest version of xPlore, a new feature can be implemented to support multiple languages in one document. Use the following configuration at the domain level to enable the feature:

```
<property value="true" name="split-text-in-mixed-lang"/><!--Enable  
splitting text for content-->  
<property value="true" name="index-metadata-in-batch"/>
```

After implementation, different language units in the same document can be analyzed by different languages.

## Phrase searches

Search fails for parts of common phrases. A common phrase like *because of, a good many*, or *status quo* is tokenized as a phrase and not as individual words. A search for a word in the phrase like *because* fails. Use the following workaround:

In the linguistic\_processor with name *rlp*, set force\_tokenize\_on\_whitespace to true.

## Stop words are case sensitive

Stop words are using exact form match (case sensitive and tense sensitive). Add all case forms for words that are not indexed, for example: *AND and And and Say, say, Said, and said*.

## Some lightweight sysobjects (LWSOs) are not fully indexed and searchable

In a Documentum Server repository, lightweight sysobjects such as emails inherit many attributes from the parent object. If the LWSOs are not materialized, a query on the inherited attributes fails. Some Documentum client applications, such as Webtop and DCO, materialize emails so they are fully searchable. SourceOne does not, so emails in SourceOne are sometimes not fully searchable.

If a query returns a result that is a lightweight sysobject (LWSO), the query does not find the object. No result is returned. DQL filters unmaterialized LWSOs, not XQuery.

## Skipped collections are not reported in query results

When a collection is unavailable or corrupted, you can make it skipped in a query so that the query can run successfully. However, xPlore does not notify you that the collection was skipped. Results can be different when the collection is restored or brought back online.

## Special characters limitations

Special characters are treated as white space. The underscore in the following document name is treated as white space. Special characters lists are limited to Unicode characters (first 65,536 code points).

Characters can be removed from the special characters list. See “[Handling special characters](#)” on page 138.

## Chinese

*Space in query causes incorrect tokenization*

A space within a Chinese term is treated in DQL as white space. A search for the string fails. For example, the term 中国 近代 is treated as 中国 AND 近代. A search for 中国近代 fails.

*Dictionary must be customized for Chinese name and place search*

For Chinese documents, the names of persons and places cannot be searched. To be found, they must be added to the Chinese dictionary in xPlore. See “[Adding dictionaries to CPS](#)” on page 152. You can also use the following workaround:

In the linguistic\_processor with name *rlp*, set generate\_cjk\_components to true and enable query with components for CJK.

## 1.3 xPlore compared to FAST

If you are migrating from FAST to xPlore, the following information describes differences between the two indexing servers.

### Administration differences

xPlore has an administration console. FAST does not. Many features in xPlore are configurable through xPlore administrator. These features were not configurable for FAST. Additionally, administrative tasks are exposed through Java APIs.

- Ports required: During xPlore instance configuration, the installer prompts for the HTTP port for the WildFly instance (base port). The installer validates that the next 100 consecutive ports are available. During index agent configuration, the installer prompts for the HTTP port for index agent WildFly instance and validates that the next 20 consecutive ports are available. FAST used 4000 ports.
- High availability: xPlore supports N+1, active/passive with clusters, and active/active shared data configurations. FAST supports only active/active. xPlore supports spare indexing instances that are activated when another instance fails. The *Documentum xPlore Installation Guide* describes high availability options for xPlore.
- Disaster recovery: xPlore supports online backup. FAST supports only offline (cold) backup.
- Storage technology: xPlore supports SAN and NAS. FAST supports SAN only.
- Virtualization: xPlore runs in VMware environments. FAST does not.
- 64-bit address space: 64-bit systems are supported in xPlore but not in FAST.

xPlore requires less temporary disk space than FAST. xPlore requires twice the index space used by all collections, in addition to the index. This space is used for merges and optimizations. FAST requires 3.5 times the space.

### Indexing differences

Back up and restore: xPlore supports warm backups.

High availability: xPlore automatically restarts content processing after a CPS crash. After a VM crash, the xPlore watchdog sends an email notification.

Transactional updates and purges: xPlore supports transactional updates and purges. FAST does not.

Collection topography: xPlore supports creating collections online, and collections can span multiple file systems. FAST does not support these features.

Lemmatization: FAST supports configuration for which parts of speech are lemmatized. In xPlore, lemmatization is enabled or disabled. You can configure lemmatization for specific Documentum attribute values.

## Search differences

One-box search: Searches from the Webtop client default to ANDed query terms in xPlore.

Query a specific collection: Targeted queries are supported in xPlore but not FAST.

Folder descend: Queries are optimized in xPlore but not in FAST.

Results ranking: FAST and xPlore use different ranking algorithms.

Excluding from index: xPlore allows you to configure non-indexed metadata to save disk space and improve ingestion and search performance. With this configuration, the number of hits differs between FAST and xPlore queries on the non-indexed content. For example, if xPlore does not index docbase\_id, a full-text search on "256" returns no hits in xPlore. The search returns all indexed documents for repository whose ID is 256.

Security evaluation: Security is evaluated by default in the xPlore full-text engine before results are returned to Documentum Server, resulting in faster query results. FAST returns results to the Documentum Server, resulting in many hits that the user is not able to view.

Underprivileged user queries: Optimized in xPlore but not in FAST.

Native XQuery syntax: Supported by xPlore.

Facets: Facets are limited to 350 hits in FAST, but xPlore supports many more hits.

Special characters: Special character lists are configurable. The default in xPlore differs from FAST when terms such as email addresses or contractions are tokenized. For example, in FAST, an email address is split up into separate tokens with the period and @ as boundaries. However, in xPlore, only the @ serves as the boundary, since the period is considered a context character for part of speech identification.

## 1.4 Architectural overview

xPlore provides query and indexing services that can be integrated into external content sources such as the Documentum content management system. External content source clients like Webtop or CenterStage, or custom Documentum DFC clients, can send indexing requests to xPlore.

Each document source is configured as a domain in xPlore. You can set up domains using xPlore administrator. For Documentum environments, the Documentum index agent creates a domain for each repository and a default collection within that domain.

Documents are provided in an XML representation to xPlore for indexing through the indexing APIs. In a Documentum environment, the Documentum index agent prepares an XML representation of each document. The document is assigned to a category, and each category corresponds to one or more collections as defined in xPlore.

xPlore instances are web application instances that reside on application servers. When an xPlore instance receives an indexing request, it uses the document category to determine what is tokenized and saved to the index. A local or remote instance of the Content Processing Service (CPS) fetches the content. CPS detects the format of a document, extracts indexable content from the document, and then performs linguistic analysis to generate tokens to be saved in index. The tokens are used for building a full-text index.

xPlore manages the full-text index. An external Apache Lucene full-text engine is embedded into the XML database (xDB). xPlore tracks indexing and update requests, recording the location of indexed content in xDB. xDB provides transactional updates to the Lucene index. Indexes are still searchable during updates.

When an instance receives a query request, the request is processed on all included collections, then the assembled query results are returned. xPlore provides a web-based administration console.

## 1.5 Physical architecture

The xPlore index service and search service are deployed as a WAR file to a wildfly application server that is included in the xPlore installer. xPlore administrator and online help are installed as war files in the same wildfly application server. The index is stored in the storage location that was selected during configuration of xPlore.

### 1.5.1 xPlore disk areas

xPlore creates disk areas for xDB data and database transaction redo log, the Lucene index within xDB, a temp area, and xPlore configuration and utilities. When you run the xPlore installer to configure an index agent, a disk area is created for content staging. The following table describes how these areas are used during indexing and search.

**Table 1-1: Disk areas for xPlore**

Area	Description	Use in indexing	Use in search
<i>xplore_home</i> /data	Stores dftxml, metrics, audit, ACLs and groups.	Index updated through inserts and merges	Random access retrieval for specific elements and summary. Inverted index lookup and facet and security retrieval
<i>xplore_home</i> /config/wal	Stores transaction information. Files in this location should not be modified without guidance from OpenText. Removal of certain files or folders can cause startup failure or malfunctions of xPlore.	Updates to xDB data are logged	Provides snapshot information during some retrievals
<i>xplore_home</i> /dsearch/backup	Used for restore when xPlore is down. The location is defined in the <i>admin-config/backup-location</i> element of <i>indexserverconfig.xml</i> .		

Area	Description	Use in indexing	Use in search
temp area	(CPS) Intermediate processingDefault location: <code>xplore_home/dsearch/cps/cps_daemon/temp</code>  (CPS) Exports to the index serviceDefault location: <code>xplore_home/dsearch/cps/cps_daemon/export</code>  Index: Updates to the Lucene index (non-transactional)Default location: <code>xplore_home/data/temp</code>	Non-committed data is stored to the log	None
Index agent content staging area	Temporarily holds content during indexing processDefault location: <code>xplore_home/&lt;wildfly_version&gt;/server/DctmServer_IndexAgentInstanceName/data/Indexagent</code>	Holds content	None

### 1.5.2 xPlore instances

An xPlore instance is a web application instance (WAR file) that resides on an application server. You can have multiple instances on the same host (vertical scaling), although it is more common to have one xPlore instance per host (horizontal scaling). Make sure those instances are installed with unique names. You create an instance by running the xPlore installer.

The first instance that you install is the primary instance. You can add secondary instances after you have installed the primary instance. The primary instance must be running when you install a secondary instance.

#### *Adding or deleting an instance*

To add an instance to the xPlore system, run the xPlore configurator script (`configDsearch.bat/configDsearch.sh` under setup folder). If an xPlore instance exists on the same host, select a different port for the new instance, because the default port is already in use.

To delete an instance from the xPlore system, use the xPlore configurator script. Shut down the instance before you delete it.

You manage instances in xPlore administrator. Click **Instances** in the left panel to see a list of instances in the right content pane. You see following instance information:

- Host information: Host, operating system, and CPU architecture.
- xPlore information: xDB version, instance version, instance type, and state.
- JVM information: JVM version, active thread count, loaded class count, JVM memory usage, process ID, and thread dump.

To view JVM memory usage information, click **Memory** and a pie chart will appear showing how much JVM memory has been used.

Click **Thread Dump** to view full Java thread dump in a pop-up window. You can click **Save as** in the window to save the thread dump data for later analysis.

An instance can have one or more of the following features enabled:

- Content processing service (CPS)
- Indexing service
- Search service
- xPlore Administrator (includes analytics, instance, and data management services)
- Spare: A spare instance can be manually activated to take over for a disabled or stopped instance. See “[Replacing a failed instance with a spare](#)” on page 40.

You manage an instance by selecting the instance in the left panel. Collections that are bound to the instance are listed on the right. Click a collection to go to the Data Management view of the collection.

The application server instance name for each xPlore instance is recorded in indexserverconfig.xml. If you change the name of the wildfly instance, change the value of the attribute *appserver-instance-name* on the *node* element for that instance. This attribute is used for registering and unregistering instances. Back up the xPlore federation after you change this file.

### 1.5.3 xDB libraries and Lucene index

xDB is a Java-based XML database that enables high-speed storage and manipulation of many XML documents. xDB supports the XQuery language and XQFT query specifications. An xDB library has a hierarchical structure like a file system directory. The library is a logical container for other libraries or XML documents.

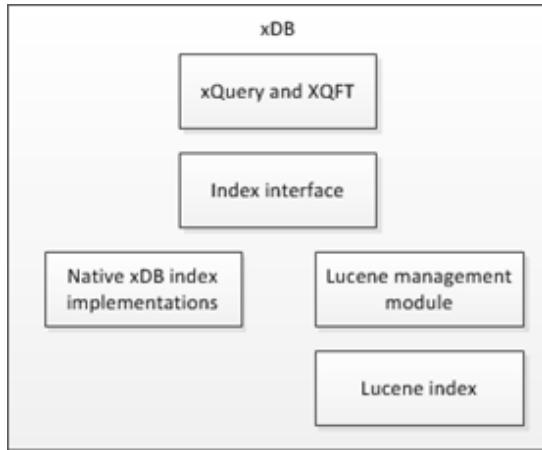
A library corresponds to a collection in xPlore with additional metadata such as category, usage, and properties. An xDB library stores an xPlore collection as one or more Lucene indexes that can include the XML content that is indexed. xPlore manages the indexes on the collection.

xDB contains the following libraries for xPlore:

- The library contains SystemData metrics and audit databases. These databases record metrics and audit queries by xPlore instance.
- Each domain library contains an xDB tracking library (database), recording the content that has been indexed, which contains several child tracking libraries. Each of these libraries is associated with a collection.
- Each domain contains one or more data libraries. The *default* library is created for a domain by default.
- Each domain may contain acl/group libraries for security filtering.

When xPlore processes an XML representation of an input document and supplies tokens to xDB, xDB stores them into a Lucene index. Optionally, xPlore can be configured to store the content along with the tokens. When documents are updated or deleted, changes to the index are propagated. When xPlore supplies XQuery expressions to xDB, which passes them to the Lucene index.

xDB manages parallel dispatching of queries to more than one Lucene index when parallel queries are enabled. For example, if you have set up multiple collections on different storage locations, you can query each collection in parallel.



**Figure 1-1: xDB and Lucene**

An xDB library is stored on a data store. If you install more than one instance of xPlore, the storage locations must be accessible by all instances. The xDB libraries can reside on separate data stores, SAN or NAS. The locations are configurable in xPlore Administrator. If you do not have heavy performance requirements, you can leave all libraries on the same data store. See ["Creating a storage location" on page 204](#).

### 1.5.4 Indexes

xDB has several possible index structures that are queried using XQuery. The Lucene index is modeled as a multi-path index (a type of composite index) in xDB. The Lucene index services both value-based and full-text probes of the index.

Covering indexes are also supported. When the query needs values, they are pulled from the index and not from the data pages. Covering indexes are used for security evaluation and facet computation.

You can configure none, one, or multiple indexes on a collection. An explicit index is based on values of XML elements, paths within the XML document, path-value combination, or full-text content. For example, the following is an XQuery using a value indexed field:

```
/dmftdoc[dmftmetadata//object_name="foo"]
```

The following is an XQuery using a tokenized, full-text field:

```
/dmftdoc[dmftmetadata//object_name ftcontains 'foo']
```

Indexes are defined and configured in `indexserverconfig.xml`. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.

## 1.6 Logical architecture

A domain contains indexes for one or more categories of documents. A category is logically represented as one or more collections. Each collection contains indexes on the content and metadata. When a document is indexed, it is assigned to a category or class of documents and indexed into one of the category collections:

- “[Documentum domains and categories](#)” on page 27
- “[Mapping of domains to xDB](#)” on page 29

### Domains

A domain is a separate, independent, logical grouping of collections within an xPlore deployment. For example, a domain could contain the indexed contents of a single Documentum content repository. Domains are defined in xPlore administrator in the data management screen. A domain can have multiple data collections in addition to the default data collection.

The Documentum index agent creates a domain for the repository to which it connects. This domain receives indexing requests from the Documentum index agent.

### Categories

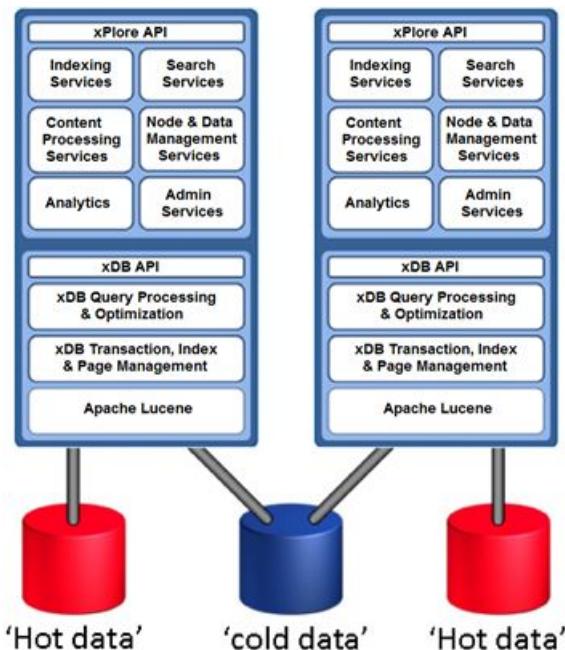
A category defines how a class of documents is indexed. All documents submitted for ingestion must be in XML format. (For example, the Documentum index agent prepares an XML version for Documentum repository indexing.) The category is

defined in indexserverconfig.xml and managed by xPlore. A category definition specifies the processing and semantics that is applied to an ingested XML document. You can specify the XML elements that are used for language identification. You can specify the elements that have compression, text extraction, tokenization, and storage of tokens. You also specify the indexes that are defined on the category and the XML elements that are not indexed. A collection belongs to one category.

## Collections

A collection is a logical group of XML documents that is physically stored in an xDB detachable library. A collection represents the most granular data management unit within xPlore. All documents submitted for indexing are assigned to a collection. A collection generally contains one category of documents. In a basic deployment, all documents in a domain are assigned to a single default collection.

A collection is bound to a specific instance in read-write state (index and search, index only, or update and search). A collection can be bound to multiple instances in read-only state (search-only). In the following figure, three collections (two hot and one cold) with their corresponding instances are shown.



**Figure 1-2: Read-write (index and search) and read-only (search-only) collections on two instances**

Use xPlore Administrator to do the following:

- Define a collection and its category
- Back up the collection

- Change the collection state to read-only or read-write
- Change the collection binding to a different instance

The metrics and audit systems store information in collections in a domain named SystemData. You can view this domain and collections in xPlore administrator. One metrics and one audit database is defined. Each database has a subcollection for each xPlore instance.

### Example

A document is submitted for indexing. The client indexing application, for example, Documentum index agent, has not specified the target collection for the document. By default, the index agent picks up one instance to process the indexing request according to the key of the request. When the instance receives the request, it checks whether the object in the request has been indexed or not. If it has already been indexed, the instance updates the object directly in the previous collection. Otherwise, the predefined collection routing mechanism is applied when there are multiple collections. If you provide a predefined routing class or the index agent provides collection hints, the default collection routing mechanism is skipped.

## 1.6.1 Documentum domains and categories

### Repository domains

An xPlore domain generally maps to a single Documentum repository. Within that domain, you can direct documents to one or more collections. In the following configuration in indexserverconfig.xml, a repository is mapped to a domain. Three collections are defined: one for metadata and content (default), one for ACLs, and one for groups. These latter two collections are used to filter results for permissions before returning them to the client application. The data collections in the domain can be distributed across multiple xPlore instances. Each collection is bound to an instance.

```
<domain storage-location-name="default" default-document-category="dftxml"
name="TechPubsGlobal">
  <collection document-category="dftxml" usage="Data" name="default"/>
  <collection document-category="dftxml" usage="acl" name="acl"/>
  <collection document-category="dftxml" usage="groups" name="groups"/>
...
</domain>
```

### Documentum categories

A document category defines the characteristics of XML documents that belong to that category and their processing. All documents are sent to a specific collection based on the document category. For example, xPlore pre-defines a category called dftxml that defines the indexes for Documentum use cases. All Documentum indexable content and metadata are sent to this category.

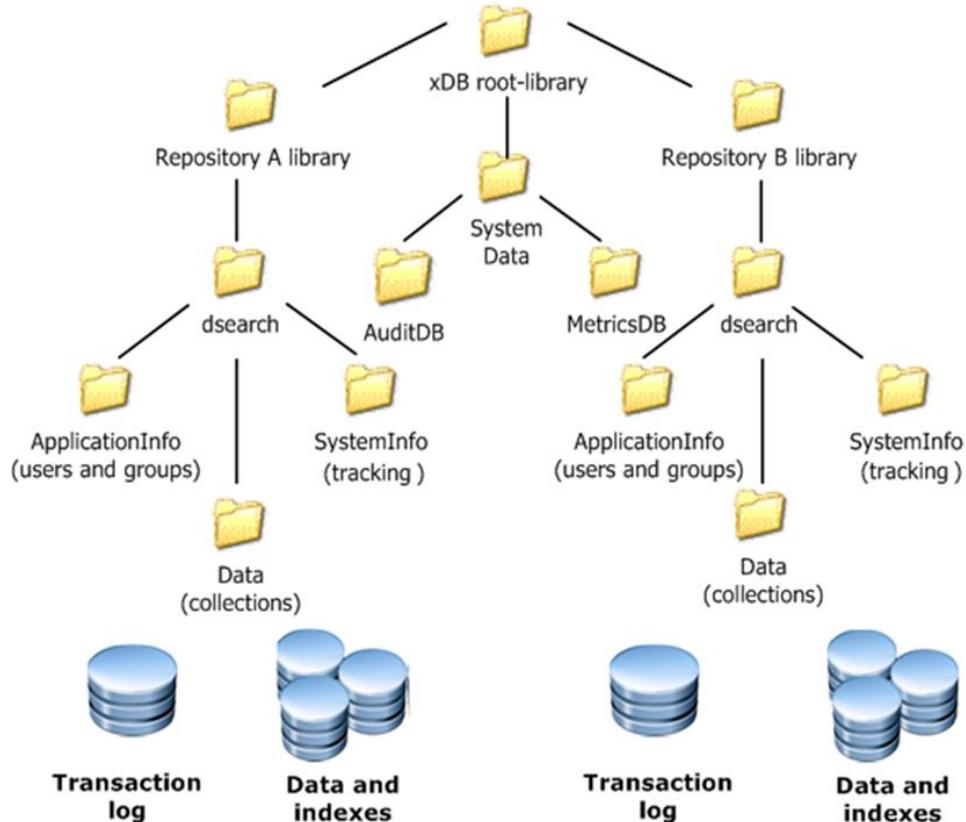
The following Documentum categories are defined within the *domain* element in indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.

- dftxml: XML representation of object metadata and content for full text indexing. To view the dftxml representation using xPlore administrator, click the document in the collection view.
- acl: ACLs that defined in the repository are indexed so that security can be evaluated in the full-text engine. See ["About security" on page 63](#) for more information.
- group: Groups defined in the repository are indexed to evaluate security in the full-text engine.



**Note:** xPlore allows only one acl and one group collection.

## 1.6.2 Mapping of domains to xDB

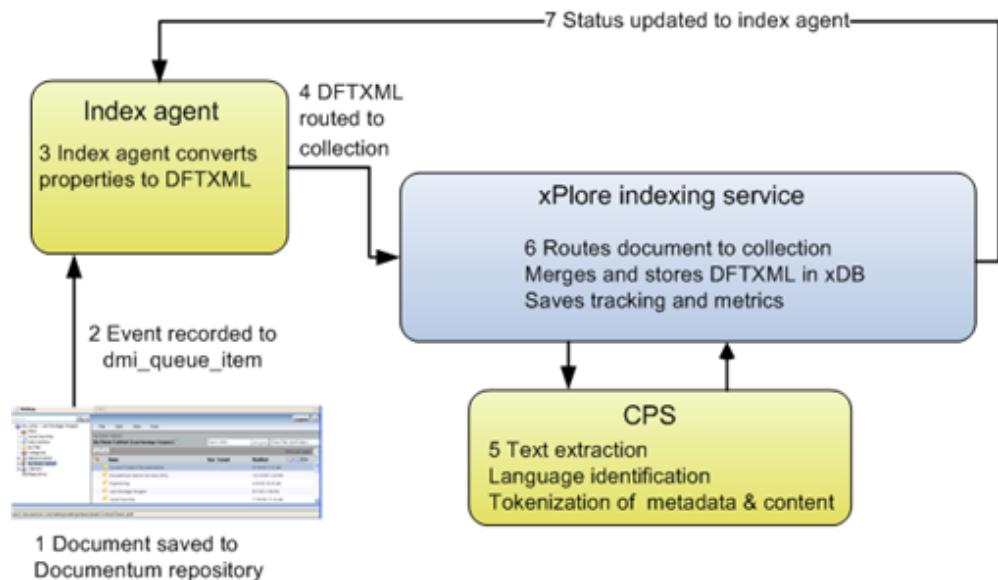


**Figure 1-3: Database structure for two instances**

- The entire xPlore federation library is stored in xDB root-library.
- One content source (Documentum repository A) is mapped to a domain library. The library is stored in a defined storage area on either instance.
- A second repository, Repository B, has its own domain.
- All xPlore domains share the system metrics and audit databases (SystemData library in xDB with libraries MetricsDB and AuditDB). The metrics and audit databases have a subcollection for each xPlore instance.
- The ApplicationInfo library contains Documentum ACL and group collections for a specific domain (repository).
- The SystemInfo library contains the collection TrackingDB. Each collection in TrackingDB matches a collection in Data and is bound to the same instance as that data collection.
- The Data library contains a default data collection.

There are also other types of libraries, for example thesaurusdb.

## 1.7 How Documentum Server documents are indexed



**Figure 1-4: xPlore indexing path**

1. In a client application, a Save, Checkin, Destroy, Readonlysave, or MoveContent operation is performed on a SysObject in the repository.
2. This operation event generates a queue item (`dmi_queue_item`) in the repository that is sent to the full-text user work queue. The full-text user, `dm_fulltext_index_user*` (`dm_fulltext_index_user`, `dm_fulltext_index_user_01`, etc.), is a Superuser created when a repository is created or when an existing repository is upgraded. The index agent retrieves the queue item and applies index agent filters. After the queue item is generated in the repository, the client application can move on to the next task. (Indexing is asynchronous.)
3. The index agent retrieves the object associated with the queue item from the repository. The content is retrieved or staged to a temporary area. The index agent then creates a `dftxml` (XML) representation of the object that can be used for full-text and metadata indexing.
4. The Index Agent sends the `dftxml` representation of the content and metadata to one instance in the xPlore deployment.
5. The xPlore indexing service calls CPS to perform text extraction, language identification, and transformation of metadata and content into indexable tokens.
6. The xPlore indexing service performs the following steps:
  - Routes documents to their target collections.

- Returns the tokens from CPS.
  - Merges the extracted content and tokens into the dftxml representation of the document. Some internal elements and metadata attributes are added to dftxml.
  - Calls xDB to store the dftxml in xDB. The analyzer associates the dftxml file with the tokens returned from CPS.
  - Stores the document location (collection) and document ID in the TrackingDB.
  - Saves indexing metrics in the MetricsDB.
  - Tracks document indexing status in the in-memory queue.
7. The indexing service notifies the index agent of the indexing status. The index agent then removes the queue item from the repository. Otherwise, the queue item is left behind with the error status and error message.

The object is now searchable. (The index service does not provide any indication that an object is searchable.) For information on how to troubleshoot latency between index agent submission and searchability, see ["Troubleshooting indexing" on page 182](#).

## Enabling indexing for an object type

Events in dmi\_registry for the user *dm\_fulltext\_index\_user\** generate queue items for indexing. The following events are registered for *dm\_fulltext\_index\_user\** to generate indexing events by default:

- dm\_sysobject: dm\_save, dm\_checkin, dm\_destroy, dm\_saveasnew, dm\_move\_content
- dm\_acl: dm\_save, dm\_destroy, dm\_saveasnew
- dm\_group: dm\_save, dm\_destroy

## Registering a type for full-text indexing

Use Documentum Administrator to change the full-text registration for an object type. Select the type, view the properties, and for the property **Enable indexing** check **Register for indexing**. To change specific events that are registered for full-text, use the DFC API registerEvent().

## Reindexing

The index agent does not recreate all the queue items for reindexing. Instead, it creates a watermark queue item (type *dm\_ftwatermark*) to indicate the progress of reindexing. It picks up all the objects for indexing in batches by running a query. The index agent updates the watermark as it completes each batch. When the reindexing is completed, the watermark queue item is updated to 'done' status.

You can submit for reindexing one or all documents that failed indexing. In Documentum Administrator, open **Indexing Management > Index Queue**. Choose

Tools > Resubmit all failed queue items, or select a queue item and choose Tools > Resubmit queue item.

## 1.8 How Documentum Server documents are queried

Several software components control full-text search using the xPlore server:

- The Documentum Server queries the full-text indexes and returns query results to client applications.
- The xPlore server responds to full-text queries from Documentum Server.

### Path of a query from a Documentum client to xPlore

1. The client application submits a DQL query or XQuery to the Documentum Server. (If the client application uses DFC 6.6 or higher to create the query, DFC translates the query into XQuery syntax.)
2. The Server transmits the query to xPlore (Documentum Server 6.6 or higher). If the query is not an XQuery, the query plugin translates it into XQuery syntax.
3. The query plugin transmits batches of HTTP messages containing XQuery statements to the xPlore search service.
4. CPS tokenizes the query based on the locale declared in the query. xDB breaks the query into XQuery clauses for full-text (using *ftcontains*) and metadata (using value constraints). The query is executed in the Lucene index against all collections unless a collection is specified in the query.
5. xDB applies the xPlore security filter to evaluate the security of the search results. If Documentum security evaluation is enabled, then security evaluation is done by the Documentum Server.
6. If a folder constraint is provided, xPlore filters documents not covered by the folder.
7. If a facet is defined in XQuery, the facet result is computed in xPlore.
8. If a summary is required in XQuery, then the summary is computed in xPlore.
9. The results are returned in batches to the query plug-in, which subsequently returns them to the client application.



# Chapter 2

## Managing the System

### 2.1 Opening xPlore Administrator

You can access the xPlore Administrator web console at one of the following locations:

```
http://host:port/dsearchadmin  
https://host:port/dsearchadmin
```

- host: DNS name of the computer on which the xPlore primary instance is installed.
- port: xPlore primary instance port (default: 9300).

To log into xPlore Administrator, do the following:

1. Log in with the xPlore Administrator user name and password. You can log in either as default superuser, with credentials you entered during installation of the primary instance, or as a user created by a superuser.

If you want to manage another xPlore deployment and it is not available for selection in the login dialog box, select **Other** to provide additional information and the xDB password.



**Note:** When you log in as a Documentum Server domain user, you must enter the domain in the login dialog box.

2. In the navigation tree, click **System Overview** to get the status of each xPlore instance, and click **Global Configuration** to configure system-wide settings. You can also navigate to the Administration page to manage user access.

### 2.2 Managing user access

To manage users in xPlore Administrator, you must log in as one of the following:

- The default administrator, with the password that you entered when installing the primary instance.
- A superuser created by the default administrator, with access to all admin console functions for an authorized xPlore deployment.

You can create and edit users, groups and deployments, assign users to groups, and select roles for groups. You can also assign users to predefined system groups that have preselected roles. Roles are predefined in the system and determine access permissions for users assigned to a group. The available roles and access permissions are shown in the following table.

**Table 2-1: xPlore Administrator roles and access permissions**

Role	Access
ROLE_SUPERUSER	All
ROLE_USER	System Overview
ROLE_REPORT	Diagnostic and Utilities/Reports
ROLE_THESAURUS	Diagnostic and Utilities/Thesaurus
ROLE_DIAGNOSTIC	Diagnostic and Utilities
ROLE_DM	Data Management
ROLE_SERVICE	Services, Instances

## 2.2.1 Creating users, groups and deployments

When creating users, you can select from available groups and deployments to provide access permissions. You can create new groups and deployments if custom access permissions are needed for a user. To create a group, deployment and user, do the following:

1. Log in with your xPlore administrator or superuser password and click **Administration**.
2. In the **Groups** tab, click **Create New Group**, enter a group name and select one or more roles to define access permissions for the group.
3. In the Deployments tab, click **Add Deployment** and enter the xDB Administrator password and values for host, port and protocol.
4. In the Users tab, click **Create New User**, enter a user name and select the password source, and then select the group and deployment you created, or any others that provide appropriate access permissions for the new user.

## 2.2.2 Changing and resetting passwords

To change your password, log in to xPlore Administrator, click your username and then click **Change Password** to open the change dialog box.

To reset a user or superuser password, log in as default administrator or superuser, click **Administration** and then edit the user settings.

To reset your default administrator password, do the following:

1. Stop the xPlore server.
2. Create `reset-password.properties` under `dsearchadmin.war/WEB-INF/classes` with the username-password pair `admin=password`, where `password` is the new default administrator password.
3. Restart the server.

## 2.3 Starting and stopping the system

If you run a stop script, run as the same administrator user who started the instance. If you are stopping the primary instance, stop all other instances first.

1. Start or stop secondary instances in xPlore administrator. Navigate to the instance in the tree and choose **Stop instance** or **Start instance**.
2. Start or stop the primary instance using the start or stop script in *xplore\_home/<wildfly\_version>/server*.

When the Wildfly java process has terminated, you see the following in *dsearch.log*:

```
<event timestamp="2013-06-11 23:34:39,066" thread="WildFly Shutdown Hook">
The XML database server is shutdown successfully.</event>
<event timestamp="2013-06-11 23:34:42,300" thread="WildFly Shutdown Hook">
The DSS instance PrimaryDsearch is shut down.</event>
```

If you did not stop secondary instances, they report a failed connection to the primary instance when you restart it.

## 2.4 Viewing and configuring global operations (all instances)

A single xPlore federation is a set of instances with a single primary instance and optional secondary instances.

1. In xPlore administrator, select **Home > System Overview** in the left panel.
2. Select a service to view the status of each instance of the service.
3. Choose **Global Configuration** to configure the following system-wide settings:
  - **Storage Location.** See “[Creating a storage location](#)” on page 204.
  - **Index Service.** See “[Document processing and indexing service configuration parameters](#)” on page 402.
  - **Search Service.** See “[Search service configuration parameters](#)” on page 407.
  - **Logging.** See “[Configuring logging](#)” on page 357.
  - **Engine.** Configure incremental backups. See “[Performing a native xDB backup in xPlore Administrator](#)” on page 217.
  - **Auditing.** See “[Auditing queries](#)” on page 302, “[Troubleshooting data management](#)” on page 209, and “[Configuring the audit record](#)” on page 48.

## 2.5 Configuring an instance

You can configure the indexing service, search service, or content processing service for a secondary instance. Select the instance in xPlore administrator and then click **Stop Instance**.

### Requirements

All instances in an xPlore deployment must have their host clocks synchronized to the primary xPlore instance host.

### Configuring the primary instance

You can set the following attributes on the primary instance element (*node*) in indexserverconfig.xml. For information on viewing and updating this file, see ["Modifying indexserverconfig.xml" on page 54](#).

- xdb-listener-port: By default, the xDB listener port is set during xPlore installation.

```
<node appserver-instance-name="primary" xdb-listener-port="19288"
primaryNode="true" status="normal" url="http://localhost:19200/dsearch/"
admin-rmi-port="19298" hostname="xxxxx" name="primary">
```
- primaryNode attribute: Set to true.
- admin-rmi-port: Specify the port at which other instances connect to xPlore administrator. By default, this value is set to the port number of the WildFly connector + 31. Default: 9331
- url: Specify the URL of the primary instance, used to set connections from additional instances.

## 2.6 Changing the host name and URL

Stop all xPlore instances.

1. Edit indexserverconfig.xml in *xplore\_home/config*. On the node element, change the values of the hostname to have the new host name and URL.
  - *xPlore\_home/config/indexserverconfig.xml*
  - *xPlore\_home/dsearch/admin/xplore.properties*
  - *xPlore\_home/dsearch/xhive/admin/xdb.bat* (windows)
  - *xPlore\_home/dsearch/xhive/admin/xdb* (linux)
  - *xPlore\_home/dsearch/xhive/admin/deletedocs.properties*
  - *xPlore\_home/dsearch/xhive/admin/query.properties*
  - *xPlore\_home/setup/indexagent/tools/ftintegrity.bat* (windows)
  - *xPlore\_home/setup/indexagent/tools/ftintegrity.sh* (linux)

- `xPlore_home/setup/indexagent/tools/aclreplication.bat` (windows)
- `xPlore_home/setup/indexagent/tools/aclreplication.sh` (linux)
- `xPlore_home/<wildfly_version>/server/DctmServer_${IndexAgent_name}/deployments/IndexAgent.war/WEB-INF/classes/indexagent.xml`
- `xPlore_home/watchdog/config/dsearch-watchdog-config.xml`
- `xPlore_home/<wildfly_version>/server/DctmServer_${xPloreInstance_name}/deployments/dsearch.war/WEB-INF/classes/indexserver-bootstrap.properties`
- `xPlore_home/<wildfly_version>/server/DctmServer_${xPloreInstance_name}/deployments/dsearch.war/WEB-INF/classes/xdb.properties`

If the IP address is changed, change the value of IP in `xPlore_home/config/XhiveDatabase.bootstrap`.

2. Primary instance only: Use iAPI to change the parameters for the host name and port in the `dm_ftengine_config` object. This change takes effect when you restart the repository.

- a. Run the following iAPI command:

```
retrieve,c,dm_ftengine_config
```

- b. Run the following iAPI command:

```
dump,c,1,
```

The host and port which may be impacted are highlighted.

```
...
param_name          [0]: dsearch_qrserver_protocol
[1]: dsearch_qrygen_mode
[2]: dsearch_qrserver_target
[3]: dsearch_qrserver_port
[4]: dsearch_config_port
[5]: dsearch_qrserver_host
[6]: dsearch_domain
[7]: dsearch_config_host
[8]: query_plugin_mapping_file
[0]: HTTP
[1]: both
[2]: /dsearch/IndexServerServlet
[3]: 9900
[4]: 9300
[5]: 10.32.122.225
[6]: DOCREPO
[7]: 10.32.122.225
[8]: D:\Documentum\fulltext
                           dm_AttributeMapping.xml
param_value
...
?
, c, select param_name, param_value from dm_ftengine_config where
r_object_id='080a0d6880000d0d'
```

- c. Enter your new xPlore port (`dsearch_qrserver_port`) at the SET command line. If the port was returned as the second parameter, set the index to 3 as shown in the following example:

```
retrieve,c,dm_ftengine_config
set,c,1,param_value[3]
```

```
SET>new_port  
save,c,1
```

- d. Enter your new xPlore host name (dsearch\_qrserver\_host) at the SET command line. For example

```
retrieve,c,dm_ftengine_config  
set,c,1,param_value[5]  
SET>new_hostname  
save,c,1
```

If the index agent is installed on the same host of xPlore, make sure dsearch\_config\_host and dsearch\_config\_port are also modified.

3. Restart the Documentum Server and xPlore instances.

## 2.7 Replacing a failed instance with a spare

You can install a spare instance using the xPlore installer. When you install a spare instance, the data, index, and log directories must all be accessible to the primary instance. Use shared storage for the spare. When you activate the spare to take over a failed instance, xPlore recovers failed data using the transaction log.

You cannot change an active instance into a spare instance.

Use xPlore administrator to activate a spare to replace a failed or stopped secondary instance. If you are replacing a primary instance, see “[Replacing a failed primary instance](#)” on page 41.

1. Stop the failed instance.
2. Open xPlore administrator and verify that the spare instance is running.
3. Select the spare instance. Click **Activate Spare Instance**.
4. Choose the instance to replace.

When xPlore administrator reports success, the spare instance is renamed in the UI with the replaced instance name. When you activate a spare to replace another instance, the spare takes on the identity of the old instance. For example, if you activated DSearchSpare to replace DSearchNode3, the spare instance becomes DSearchNode3. The old instance can no longer be used for ingestion or queries. The failed instance is renamed with Failed appended, for example, DSearchNode3Failed.

The activated instance is not registered with the watchdog service. To register it for watchdog notifications, edit the configuration file dsearch-watchdog-config.xml. This file is located in *xplore\_home/watchdog/config*.

1. Copy and paste an existing watchdog-config element for an active instance.
2. Edit the following properties for the activated spare in your copied watchdog-config element:

```
watchdog-config[@host-name]  
watchdog-config/application-config[@instance-name]
```

```
watchdog-config/application-config/properties/property[
  @name="application_url" value]
watchdog-config/application-config/tasks/task[@category="process-control" id]
```

3. Restart the watchdog service (Windows) or run the watchdog script (Linux).

For information on changing a failed instance to spare, see “[Changing a failed instance into a spare](#)” on page 43.

## 2.8 Replacing a failed primary instance

1. Shut down all xPlore instances. The shutdown scripts are located in *xplore\_home/<wildfly\_version>/server*. (On Windows, each instance is installed as an automatic service.) If you run a stop script, run as the same administrator user who started the instance.
2. Edit *indexserverconfig.xml*, which is located in *xplore\_home/config*.
3. Locate the spare *node* element in *indexserverconfig.xml*. (The status attribute is set to *spare*.)
  - a. Set the status to *normal*.
  - b. Change the value of the *primaryNode* attribute to *true*.
  - c. Change the value of the *name* attribute to the name of your previous primary instance, for example, *PrimaryDsearch*. Use the previous primary instance name.



**Note:** Do not change the value of *<appserver-instance-name>*.

4. Locate the *node* element for the old primary instance. Note the name of the node for the next step. Delete this node element.
5. Validate your changes using the validation tool described in “[Modifying indexserverconfig.xml](#)” on page 54.
6. Edit *indexserver-bootstrap.properties* in the web application for the new primary instance, for example, *xplore\_home/<wildfly9.0.1>/server/DctmServer\_Spare/deployments/dsearch.war/WEB-INF/classes*.
  - a. Change the value of the *node-name* property to *PrimaryDsearch*.
  - b. Change the value of the *isPrimary* property to *true*.
  - c. Change the value of *xhive-connection-string* to match the host and port of your new primary instance, for example:
 

```
xhive-connection-string=xhive\:///10.32.168.105\:9432
```
  - d. Edit *indexserver-bootstrap.properties* in all other xPlore instances and change the value of *xhive-connection-string* to connect to the new primary instance.
7. Edit *xdb.properties* in the directory WEB-INF/classes of the new primary instance.

- a. Find the XHIVE\_BOOTSTRAP entry and edit the URL to reflect the new primary instance host name and port. (This bootstrap file is not the same as the indexserver bootstrap file.)
  - b. Change the host name to match your new primary instance host.
  - c. Change the port to match the port for the value of the attribute xdb-listener-port on the new instance. For example:

```
XHIVE_BOOTSTRAP=xhive://NewHost:9830
```
  - d. Edit xDB.properties in all other xPlore instances to reference the new primary instance.
8. Update xdb.bat in *xplore\_home/dsearch/xhive/admin*. Your new values must match the values in *indexserverconfig.xml* for the new primary instance.
    - Change the path for XHIVE\_HOME to the path to the new primary instance web application.
    - Change ESS\_HOST to the new host name.
    - Change ESS\_PORT to match the value of the port in the url attribute of the new primary instance (in *indexserverconfig.xml*).
  9. Start the xPlore primary instance, then start the secondary instances.
  10. Update the index agent.
    - a. Shut down the index agent instance and modify *indexagent.xml* in *xplore\_home/<wildfly\_version>/server/DctmServer\_Indexagent/deployments/IndexAgent.war/WEB-INF/classes*.
    - b. Change parameter values for parameters that are defined in the element *indexer\_plugin\_config/generic\_indexer/parameter\_list/parameter*.
      - Change the *parameter\_value* of the parameter *dsearch\_config\_host* to the new host name.
      - Change the *parameter\_value* of the parameter *dsearch\_config\_port* to the new port.
  11. Update these properties of *dm\_ftengine\_config* on the Documentum Server: *dsearch\_qrserver\_host*, *dsearch\_qrserver\_port*, *dsearch\_config\_host*, and *dsearch\_config\_port*. Use iAPI to change the parameters for the host name and port in the *dm\_ftengine\_config* object. This change takes effect when you restart the repository.
    - a. To find the port and host parameter index values for the next step, do the following iAPI command:

```
retrieve,c,dm_ftengine_config
```
    - b. Use the object ID to get the parameters and values and their index positions. For example:

```
? ,c,select param_name, param_value from dm_ftengine_config where
r_object_id='080a0d6880000d0d'
```

- c. To set the port, enter your new port at the SET command line. If the port was returned as the third parameter in step 3, substitute 3 for the parameter index. For example:

```
retrieve,c,dm_ftengine_config
set,c,1,param_value[3]
SET>new_port
save,c,1
```

- d. To set the host name, enter your new host name at the SET command line:

```
retrieve,c,dm_ftengine_config
set,c,1,param_value[4]
SET>new_hostname
save,c,1
```

## 12. Back up the federation.

### *Troubleshooting primary failover*

- If you did not configure xPlore administrator when you set up the spare, extract dsearchadmin.war from *xplore\_home/setup/dsearch* into your spare instance deployment subdirectory in <*wildfly\_version*>. Update the path to dsearchadminweb.log in logback.xml of the spare instance. Specify a path to the logs directory in the primary instance WildFly application on the spare instance.
- A startup error FEDERATION\_ALREADY\_OPEN can be encountered when the old primary instance has not fully terminated before you replace it. The best way to confirm instance shutdown on Windows is to set the Windows xPlore service to manual. Use the WildFly script to stop the instance. If you must use the Windows service, check dsearch.log on the old primary instance to find “The xPlore instance PrimaryDsearch shutdown is complete.” Then configure the new primary instance.

## 2.9 Changing a failed instance into a spare

Because the identity of a failed instance is assigned to another instance, the identity of the failed instance must be changed.

1. Open indexserverconfig.xml in *xplore\_home/config*.
  - a. Change the *node* element *name* attribute of the failed instance to a new unique name.
  - b. Change the *node* element *status* attribute to *spare*.
2. Modify indexserver-bootstrap.properties in the WEB-INF/classes directory of the application server instance, for example:  
*xplore\_home/<wildfly9.0.1>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes*  
 Change the node-name key value to the one you set in indexserverconfig.xml.

3. Restart the primary and then the secondary instances.
4. Back up the xPlore federation.

## 2.10 Starting and stopping the watchdog service

The xPlore watchdog service is a Windows service or daemon process that monitors and checks the status of various processes in xPlore. One watchdog service is installed on each xPlore host. If a host has multiple xPlore instances, the watchdog service can monitor all instances. If a process such as the indexing or search service fails, the watchdog service detects the failure and sends an email notification to the administrator.

When an xPlore instance or index agent instance is deleted, the watchdog configuration file is updated to remove that instance. If you activate a spare xPlore instance, you must manually configure the watchdog service as described in “Replacing a failed instance with a spare” on page 40.

On Windows hosts, the watchdog process starts at xPlore installation and when the host is booted up. On Linux hosts, you must start the watchdog process manually. It runs as a standalone Java process.

1. To turn off the watchdog service:
  - On Windows hosts, stop the watchdog service: Documentum Search Services Watchdog.
  - On Linux hosts, run the script `stopWatchdog.sh` in `xplore_home/watchdog`. If you run a stop script, run as the same administrator user who started the instance.
2. To restart the watchdog service:
  - On Windows hosts, start the watchdog service: Documentum Search Services Watchdog.
  - On Linux hosts, run the script `startWatchdog.sh` in `xplore_home/watchdog`.

## 2.11 Configuring the watchdog service

To configure watchdog timing, edit the configuration file `dsearch-watchdog-config.xml`. This file is located in `<xplore_home>/watchdog/config`. The following timing properties within the element `timing-infocan` be configured:

- recurrence timeunit and frequency: Specifies how often the task is executed. For example, the disk space task with a frequency of 2 and time unit of hours checks disk space every two hours. Default: Every minute.
- start-date: date and time the task should be invoked, in UTC format. If the date is in the past, the task will be executed as soon as possible.
- expiry-date: Specifies the date and time a task stops executing, in UTC format.

- max-response-timeout: Specifies how long between detection of a hung task and execution of the notification (or other task). For example, a wait-time value of 6 and time unit of hours indicates a wait of 6 hours before notification about a non-responding instance.
- max-retry-threshold: Specifies the maximum number of times the task can be retried. For example, if the task is notification, a value of 10 indicates the notification task is retried 10 times. Recurring tasks are retried at the next scheduled invocation time.
- max-iterations: Maximum number of times to attempt to ping an instance that has no response. Default: -1 (no limit)

You can configure the watchdog service to monitor the index agent and either sends a notification to the administrator or restart the index agent if it is found not running, depending on your configuration.

When an index agent instance is registered to the watchdog service, the index agent task information is written to the watchdog configuration file `dsearch-watchdog-config.xml` located in `<xplore_home>/watchdog/config`, similar to the following:

```

<application-config instance-name="usesbatchmd4c_9200_IndexAgent" name="IndexAgent">
  <properties>
    <property name="application_url" value="http://usesbatchmd4c:9200/IndexAgent"/>
    <property name="docbase_user" value="batchm"/>
    <property name="docbase_name" value="xplore1"/>
    <property name="docbase_password" value="L5hMU5Ir7hmG0QxmRqJ2Dw=="/>
    <property name="servlet_wait_time" value="3000"/>
    <property name="servlet_max_retry" value="5"/>
    <property name="action_on_servlet_if_stopped" value="restart"/>
  </properties>
  <tasks>
    <task category="process-control"
      id="usesbatchmd4c_9200_IndexAgent_HeartBeat">
      <task-info>
        <primary-handler id="FtIndexAgentRestartTask"
          impl_class="com.emc.ess.watchdog.common.impl.tasks.custom.FtIndexAgentRestartTask"/>
        <failure-handler
          impl_class="com.emc.ess.watchdog.common.impl.tasks.SendFailureNotificationTask">
          <properties refid="SendMailTask"/>
        </failure-handler>
      </task-info>
      <timing-info>
        <recurrence time-unit="minutes" frequency="1"/>
        <start-date>2010-12-08T20:21:00.843Z</start-date>
        <expiry-date>2299-12-08T20:21:00.843Z</expiry-date>
        <max-response-timeout time-unit="seconds" wait-time="-1"/>
        <max-retry-threshold>5</max-retry-threshold>
      </timing-info>
    </task-info>
    </task>
  </tasks>
</application-config>

```

You can configure the timing properties for the index agent just as for other components. In addition, you can modify the following properties:

- docbase\_password: Installation owner password in encrypted format. This value is set during the index agent configuration process. If you change the installation

owner password, change this to the new encrypted password. To encrypt the password, run `<xplore_home>/watchdog/tools/encrypt-password.bat|sh`.

- `servlet_wait_time`: Specify in milliseconds the amount of time the watchdog service waits before checking the status of the index agent servlet again if the servlet is already in the process of shutting down. Default: 3000.
- `servlet_max_retry`: The number of times the watchdog service waits and check the status of the index agent servlet while the servlet is shutting down. Default: 5.
- `action_on_servlet_if_stopped`: Specify what action the watchdog service takes when it detects that the index server is not running:
  - `notify`: Send a notification email to the registered email address. This is the default value.
  - `restart`: Automatically start the index agent servlet.
  - `none`: Do nothing.

## 2.12 Configuring disk space monitoring

The watchdog can detect out of space issues in xPlore instances and monitors storage locations and the config and log directories. You can configure the watchdog to send a notification to the administrator or to stop the index agent when the limit of available disk space is reached.

To configure watchdog disk space monitoring, edit the configuration file `dsearch-watchdog-config.xml`. This file is located in `xplore_home/watchdog/config`. The task ID is similar to `PrimaryDsearch_DiskFreeSpaceMonitor`. The ID changes depending on the ID of the xPlore instance. The following properties can be configured. All but the first two properties are configured within the `timing-info` element.

- `stop_index_agent_when_out_of_disk_space`: Specifies that the index agent stops at a specified percentage of available disk space. Default: true.
- `percent_available_space_to_take_action`: Value attribute: Percentage of free disk space. Default: 30.
- `recurrence_timeunit` and `frequency`: Specifies how often disk space is checked. For example, a frequency of 2 and time unit of hours checks every two hours. Default: 120 minutes (2 hours).
- `start-date`: date and time the task should be invoked, in UTC format. If the date is in the past, the task will be run as soon as possible.
- `expiry-date`: Specifies the date and time a task stops executing, in UTC format.
- `max-response-timeout`: Specifies how long between detection of a hung task and execution of the notification (or other task). For example, a wait-time value of 6 and time unit of hours indicates a wait of 6 hours before notification about a non-responding instance.
- `max-retry-threshold`: Specifies the maximum number of times the task can be retried before the next scheduled invocation time.

- max-iterations: Not applicable to this task. Default: -1 (no limit)

When the data in a remote shared environment is referenced by a local symbolic link in Windows, the watchdog cannot monitor the disk space of the remote environment.

## 2.13 Configuring xDB log monitoring

The watchdog can monitor xDB wal (write ahead log) folder changes. If the xDB log sequence number (LSN) is not updated when the xDB log changes (this can be caused by an xDB internal defect, a long transaction, or other reasons), the watchdog sends a notification to the administrator.

To configure watchdog xDB log monitoring, edit the configuration file dsearch-watchdog-config.xml. This file is located in *xplore\_home/watchdog/config*. The task ID is similar to PrimaryDsearch\_XDBLogMonitor. The ID changes depending on the ID of the xPlore instance. The following properties can be configured. All but the first property are configured within the timing-info element.

- xdb\_log\_folder\_size\_increase\_threshold: Specifies increase volume in bytes.

Watchdog checks the xDB log folder size at an interval based on the recurrence property of timing-info. If the increase volume does not reach the specified threshold, no notification is sent even when the LSN is not updated upon the xDB log change. For example, if the interval is 10 minutes and this property is set to 100, notifications can be sent only when the xDB log folder size grows by 100 bytes or more within 10 minutes.

By default, this property is set to 0, meaning that notifications are sent when the LSN is not updated upon xDB log growth in any size.

- recurrence timeunit and frequency: Specifies how often the xDB log folder is checked. For example, a frequency of 2 and time unit of hours checks every two hours. Default: 10 minutes.
- start-date: date and time the task should be invoked, in UTC format. If the date is in the past, the task will be executed as soon as possible.
- expiry-date: Specifies the date and time a task stops executing, in UTC format.
- max-response-timeout: Specifies how long between detection of a hung task and execution of the notification (or other task). For example, a wait-time value of 6 and time unit of hours indicates a wait of 6 hours before notification about a non-responding instance.
- max-retry-threshold: Specifies the maximum number of times the task can be retried before the next scheduled invocation time.
- max-iterations: Not applicable to this task. Default: -1 (no limit)

## 2.14 Configuring system metrics

Configure system metrics in indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54. For information on the settings for indexing and CPS metrics, see “[Document processing and indexing service configuration parameters](#)” on page 402.

By default, system metrics are saved in batches of 100 every 5 seconds. To change these values, add the following line to the *system-metrics-service* element. The wait-timeout unit is seconds. For example, if wait-timeout is set to 10, the latest metrics are available about 10 seconds later (average 5 seconds). The batch size determines how many metrics are accumulated before they are saved to the system metrics database in xDB. If batch size is reached before timeout, the batch is recorded.

```
<persistence-service batch-size="100" wait-timeout="10"/>
```

When setting the values of the following task intervals inside the *system-metrics-service* element, you must specify DAYS as the time unit, rather than MINUTES, SECONDS, or MILLISECONDS; otherwise, the frequent deletion might result in additional load on the system.

```
<system-metrics-service enable="true">
  <task interval="60" enable="true" name="INDEX"/>
  <task interval="3600" enable="true" name="PURGE">
    <properties>
      <property value="90" name="delete-older-than"/>
      <property value="DAYS" name="units"/>
    </properties>
  </task>
</system-metrics-service>
```

## 2.15 Configuring the audit record

In most cases, you do not need to modify the default configuration in indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.

Records over 30 days old are automatically purged. You can configure the purge schedule:

- *auditing/location* element: Specifies a storage path for the audit record. Attributes: name, path, size-limit. Size limit units: K | M | G | T (KB, MB, GB, TB). Default: 2G
- *audit-config* element: Configures auditing. Attributes: component, status, format, location.



**Note:** Changes to format and location are not supported in this release.

- *properties/property* element:
  - *audit-save-batch-size*: Specifies how many records are batched before a save. Default: 100.

- *lifespan-in-days*: Specifies time period before audit record is purged. Default: 30
- *preferred-purge-time*: Specifies the time of day at which the audit record is purged. Format: hours:minutes:seconds in 24-hour time. Default: midnight (00:00:00)

## Viewing the audit record

The entire audit record can be large, and viewing can cause an out of memory error. Use reports to query audit records. Reports allow you to specify a time period and other report criteria. You can customize reports to query the audit record.

To view the entire audit record, drill down to the AuditDB collection in Data Management > SystemData. Click AuditDB and then click auditRecords.xml. An audit record has the following format in XML:

```
<event name=event_name component=component_name timestamp=time>
  <element-name>value</element-name>
  ...
</event>
```

## 2.15.1 Anonymizing Search Users

When auditing is enabled for search, each query performed by the user is logged as an audit record in AuditDB, recording detailed information such as who performed the query on which collection and when. You can view these records using the Audit Records for Search Component report in xPlore Administrator.

However, sometimes you do not want users to be identified in their search activities out of considerations for online privacy or security concerns so that this information is not subject to potential exploitation. You can configure the system to not record actual search usernames when logging query audit records. To anonymize search users in audit records, stop the xPlore instance and edit indexserverconfig.xml to add the anonymize-user-name property under the auditing element and set its value to true.

```
<security>
  <auditing status="on">
    <audit-config component="search" status="on" format="xml"
      location="default"/>
    <audit-config component="admin" status="on" format="xml"
      location="default"/>
    <properties>
      <property value="100" name="audit-save-batch-size"/>
      <property value="30" name="lifespan-in-days"/>
      <property value="00:00:00" name="preferred-purge-time"/>
      <property value="true" name="anonymize-user-name"/>
    </properties>
  </auditing>
</security>
```

After the new setting takes effect, all query events will still be logged but only an anonymous user *unknown* will be stored in each audit record. All audit records before the change will not be affected. Turning on the anonymize-user-name setting also has the following additional impact:

- Since query warmup is typically done by replaying the most recent queries (number\_of\_queries\_per\_user) from the last N users (number\_of\_unique\_users) loaded from the audit log, anonymizing search users renders user-based settings in the query warm-up configuration ineffective. Also, security evaluation does not work for query warm-up when search users are not logged in audit records.

If you still want to take advantage of the query warm-up feature, you need to tune the following settings in `query.properties`:

- Remove or comment out this line:

```
exclude_users=unknown
```

- Set the number of logged queries you want to replay in the `number_of_queries_per_user` property.

- You cannot use user name as an effective filter criteria when viewing the following reports:

- User Activity Report
- Detailed Query Analysis
- QBS Activity Report by Id
- Query Counts by User
- Top N Slowest Queries
- User Activity Report

## 2.16 Troubleshooting system problems

For troubleshooting installation, see *Documentum xPlore Installation Guide*.

### Checking the installation versions

All xPlore instances and index agents should have the same installation version. You can check the version in the `version.properties` file in `xplore_home/installinfo`.

### Connection refused

Indexing fails when one of the xPlore instances is down. The error in `dsearch.log` is like the following:

```
CONNECTION_FAILED: Connect to server at 10.32.112.235:9330 failed,  
Original message:  
Connection refused
```

Check the following causes for this issue:

- *Instance is stopped*: A query that hits a document in a collection bound to the stopped instance fails. For example, you create collection2 and upload a file to it. You set the mode for collection2 to `search_only` and bind the collection to instance2 and instance3. Stop instance2 and query for a term in the uploaded file.

- *The xPlore host name has changed:* If you have to change the xPlore host name, do the following:
  1. Update `indexserverconfig.xml` with the new value of the URL attribute on the `node` element. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.
  2. Change the WildFly startup (script or service) so that it starts correctly. If you run a stop script, run as the same administrator user who started the instance.
  3. Modify `indexserver-bootstrap.properties` under `<xplore_home>/<wildfly_version>/server/DctmServer_<serverName>/deployments/dsearch.war/WEB-INF/classes` to update `xhive-connection-string` with the new host name.

## Unable to start WildFly

If the file system read/write permissions have changed, you see an error like the following:

```
ERROR setFile(null,false) call failed.  
java.io.FileNotFoundException: ...boot.log (Read-only file system)
```

## Timing problems: Login ticket expired

All instances in an xPlore deployment must have their host clocks synchronized to the primary xPlore instance host. Shut down all xPlore instances, synchronize clocks, and restart.

## Which indexing engine are you using?

xPlore is represented in the repository by the ft engine config object (`dm_ftengine_config`). There is one instance of the `dm_ftengine_config` object for each full-text index engine object. If there is no `dm_ftengine_config` object, then full-text is not enabled in the Documentum Server. When you configure an index agent for a repository, full-text is automatically enabled.

*Verify full-text enabled*

Use the following iAPI command to get the object ID of the indexing engine:

```
retrieve,c,dm_ftengine_config
```

The `dm_fulltext_index` object attribute `is_standby` must be set to false (0). Substitute your object ID:

```
retrieve,c,dm_fulltext_index  
3b0012a780000100  
?,c,select is_standby from dm_fulltext_index where r_object_id='3b0012a780000100'  
0
```

*Verify which engine is used*

To verify that the xPlore engine in Documentum Server is being used, get the `object_name` attribute of the `dm_ftengine_config` object after you have retrieved the

object. Substitute the object ID in the following command. *DSearch Fulltext Engine Configuration* is returned for xPlore. The FAST configuration object name contains *FAST*:

```
retrieve,c,dm_ftengine_config  
080012a780002900  
?,c,select object_name from dm_ftengine_config where r_object_id='080012a780002900'  
DSearch Fulltext Engine Configuration
```

## I/O errors, or no such file or directory

The following causes can result in this error:

- *Multiple instances of xPlore*: Storage areas must be accessible from all other instances. If not, you see an I/O error when you try to create a collection. Use the following cleanup procedure:
  1. Shut down all xPlore instances.
  2. Edit `xhivebootstrap` in `xplore_home/config`. Change the binding node value to *primary* for segments that have this problem.
  3. Edit `indexserverconfig.xml` to remove binding elements from the collection that has the issue. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.
  4. Restart xPlore instances.
- *I/O error indexing a large collection*: Switch to the 64-bit version of xPlore and use 4+ GB of memory when a single collection has more than 5 million documents.
- *I/O error during index merge*: Documents are added to small Lucene indexes within a single collection. These indexes are merged into a larger final index to help query response time. The final merge stage can require large amounts of memory. If memory is insufficient, the merge process fails and corrupts the index. Allocate 4 GB of memory or more to the JVM.

```
com.xhive.error.XhiveException: IO_ERROR:  
Failure while merging external indexes, Original message:  
Insufficient system resources exist to complete the requested service
```

To fix a corrupted index, see “[Repairing a corrupted index](#)” on page 222. To delete a corrupted domain, see “[Delete a corrupted domain](#)” on page 193.

## High-volume/low memory errors

- *Application server out of memory*  
`java.lang.OutOfMemoryError`: GC overhead limit exceeded. Increase the default and maximum JVM heap size.
  1. Stop the application server.
  2. Edit the script that launches an xPlore server or index agent, located in `xplore_home/<wildfly_version>/server`.
  3. Increase the values of `-Xms` and `-Xmx`, save, and restart the application server.

- 4. Windows: Each instance is installed as an automatic service. Stop the service, edit the launch script, and restart the service.
- *Virtual environment:* Virtual environments are sometimes under powered or other applications in the environment are overusing the resources. You can move to a less-used virtual host or add more memory or cores to the virtual host.

## Error on startup

Non-ASCII characters in indexserverconfig.xml can cause startup to fail. If you edit indexserverconfig.xml using a simple text editor like Notepad, non-ASCII characters such as ü are saved in native (OS) encoding. For example, Windows uses ISO-8859-1. xPlore uses UTF-8 encoding, which results in unexpected text errors. Use an XML editor to edit the file, and validate your changes using the xplore.bat (Windows) or xplore.sh (Linux) script in *xplore\_home/dsearch/xhive/admin*. Restart the xPlore instances.

## Cannot change binding on a stopped instance

If an xPlore instance is stopped or crashed, you cannot change the binding of collections on that instance to another instance. Do one of the following:

- Restart before changing the binding.
- If the instance has crashed, reboot the server before changing binding. If you restart the application server without reboot, updated pages are not written to the disk.

## Timeout error of an indexing request

When an index request times out, xPlore cancels it. The error in dsearch.log is like the following:

```
operation-id of r_object_id failed to be processed in 3600000 ms,
and will be cancelled.
```

If the indexing is very slow, you can modify the timeout for a request. The *index-request-time-out* property specifies in milliseconds how much time is allowed before an index request times out. The default value is 3600000 (1 hour). To set the *index-request-time-out* parameter, add it to indexserverconfig.xml under *index-config/properties* element and specify its value. Remove this parameter from indexserverconfig.xml to use its default value. Here is an example for a two-hours setting:

```
<property name="index-request-time-out" value="7200000" />
```

## NotSerializableException warning during index agent startup or shutdown

When you see a NotSerializableException warning in *server.log* after running the index agent startup or shutdown script, it means that WildFly session persistence is enabled. This exception does not impact any index agent functionality except for making the startup or shutdown process a little longer.

You can either ignore this exception warning or perform the following steps to disable WildFly session persistence:

1. Shut down the index agent.
2. Edit the file `standalone.xml` in `<xplore_home>\wildfly_version\server\DtcmServer_Indexagent`.

Change the following paragraph:

```
<subsystem xmlns="urn:wildfly:domain:undertow:2.0">
    <buffer-cache name="default"/>
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket=
            "https" max-post-size="0"/>
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content"/>
            <filter-ref name="server-header"/>
            <filter-ref name="x-powered-by-header"/>
        </host>
    </server>
    <servlet-container name="default">
        <jsp-config/>
        <persistent-sessions path="d:/temp/session-dump"/>
        <websockets/>
    </servlet-container>
    <handlers>
        <file name="welcome-content" path="${wildfly.home.dir}/
            welcome-
            content"/>
    </handlers>
    <filters>
        <response-header name="server-header" header-name="Server"
            header-
            value="WildFly/9"/>
        <response-header name="x-powered-by-header"
            header-name="X-
            Powered-By" header-value="Undertow/1"/>
    </filters>
</subsystem>
```

Remove the `persistent-sessions` element.

3. Start the index agent.

## 2.17 Modifying indexserverconfig.xml

Some tasks are not available in xPlore administrator. These rarely needed tasks require manual editing of `indexserverconfig.xml`. This file is located in `xplore_home/config` on the primary instance. It is loaded into xPlore memory during the bootstrap process, and it is maintained in parallel as a versioned file in xDB. All changes to the file are saved into the xDB file at xPlore startup.

On Windows 2008, you cannot save the file with the same name, and the extension is not shown. By default, when you save the file, it is given a .txt extension. Be sure to replace `indexserverconfig.xml` with a file of the same name and extension.



**Note:** Do not edit this file in xDB, because the changes are not synchronized with xPlore.

1. Stop all instances in the xPlore federation.
2. Make your changes to `indexserverconfig.xml` on the primary instance using an XML editor. Changes must be encoded in UTF-8. Do not use a simple text editor

such as Notepad, which can insert characters using the native OS encoding and cause validation to fail.

3. Set the configuration change check interval as the value of *config-check-interval* in milliseconds on the *rootindex-server-configuration* element. The system will check for configuration changes after this interval.
4. Validate your changes using the CLI validateConfigFile. From the command line, type the following. Substitute your path to indexserverconfig.xml using a forward slash. Syntax:

For example:

```
xplore validateConfigFile path_to_config_file
```

For example:

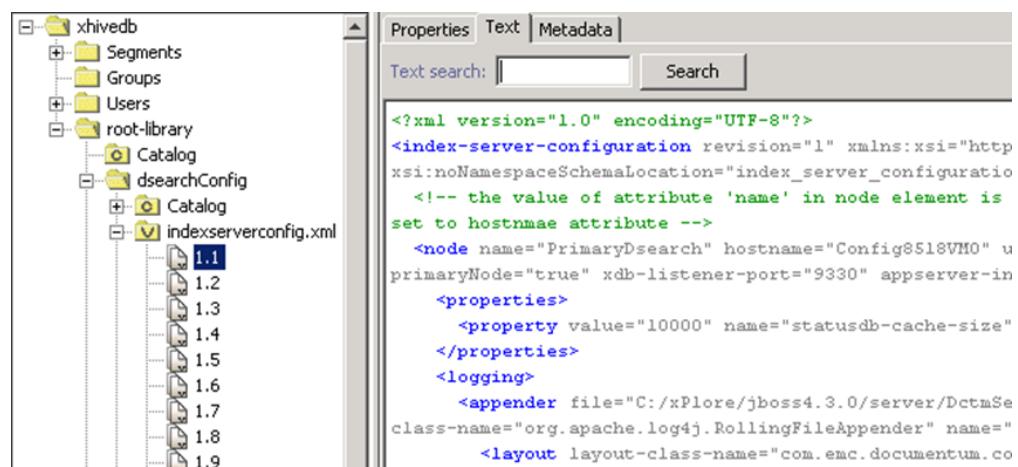
```
xplore "validateConfigFile 'C:/xPlore/config/indexserverconfig.xml'"  
./xplore.sh "validateConfigFile '/root/xPlore/config/indexserverconfig.xml'"
```

5. Back up the xPlore federation after you change this file.
6. Restart the xPlore system to see changes. The configuration file on the file system is compared on startup to the one in database even if the revision number is the same.

When indexserverconfig.xml is malformed, xPlore cannot start.

#### Troubleshooting

You can view the content of each version using the xadmin tool (see “[Debugging queries](#)” on page 318). Drill down to the *dsearchConfig* library, click a version, and then click **Text**:



**Figure 2-1: XML content in xDB admin**

### 2.17.1 Customizations in indexserverconfig.xml

- Define and configure indexes for facets.
- Add and configure categories: Specifying the XML elements that have text extraction, tokenization, and storage of tokens. Specify the indexes that are defined on the category and the XML elements that are not indexed. Change the collection for a category.
- Configure system, indexing, and search metrics.
- Specify a custom routing-class for user-defined domains.
- Change the xDB listener port and admin RMI port.
- Turn off lemmatization.
- Lemmatize specific categories or element content.
- Configure indexing depth (leaf node).
- Change the xPlore host name and URL.
- Set the security cache size.

Changes to the following configurations require an index rebuild to take effect.

- Add or change special characters for CPS processing:

```
<content-processing-services analyzer="rlp" context-characters="! ,;?" special-  
characters="@#$%^~`&.:() -+=>' /[]{}">
```

- Boost metadata and freshness in results scores:

```
<property value="0.5" name="freshness-weight"/>
```

- Configure xPlore to match phrases exactly in queries.

```
<search-config>  
  <properties>  
    <property value="en" name="query-default-locale"/>  
    ...  
    <property value="true" name="query-exact-phrase-match">  
  </properties>  
</search-config>
```

## 2.18 Tasks performed outside xPlore administrator

Some xPlore administration tasks can be performed in xPlore administrator as well as in indexserverconfig.xml or xDB. Use xPlore administrator for those tasks. (The following tasks are not common.)

**Table 2-2: Tasks outside xPlore administrator**

Action	indexserverconfig.xml	xDB
Define/change a category of documents (see “Configuring categories” on page 194)	X	

Action	indexserverconfig.xml	xDB
Define a subpath for facets (see “Configuring facets in xPlore” on page 334)	X	
Disable system, index, search metrics (see “Configuring system metrics” on page 48)	X	
Register custom routing class (see <i>Documentum Search Development Guide</i> )	X	
Change primary instance or xPlore host (see “Replacing a failed primary instance” on page 41)	X	
Configure lemmatization (see “Configuring query lemmatization” on page 257)	X	
Configure indexing depth (see “Configuring text extraction” on page 171)	X	
Boost metadata and freshness (see “Configuring scoring and freshness” on page 246)	X	
Change special characters list (see “Handling special characters” on page 138)	X	
Configure Documentum security properties (see “Changing search results security” on page 63)	X	

## Index agent tasks in the Documentum environment

The following index agent tasks are performed outside the index agent UI:

- Limit content size for indexing, see “Maximum document and text size” on page 124.
- Exclude ACL and group attributes from indexing, see “Configuring the index agent after installation” on page 81.
- Map file stores in shared directories, see “Sharing content storage” on page 88.
- Install an additional index agent for ACLs and groups, see “Setting up index agents for ACLs and groups” on page 80.
- Map partitions to specific collections, see “Mapping Server storage areas to collections” on page 90.

- Verify index agent migration, see “[Using ftintegrity](#)” on page 94.
- Customize indexing and query routing, filter object types, and inject metadata, see “[Route a document to a collection](#)” on page 187, “[Configuring Index Agent filters](#)” on page 85, and “[Injecting data and supporting joins](#)” on page 108.

## Search configuration tasks in the Documentum environment

The following search configuration tasks are performed outside xPlore administrator.

- Turn off xPlore native security, see “[Changing search results security](#)” on page 63.
- Make types and attributes searchable, see “[Making types and attributes searchable](#)” on page 274.
- Turn off XQuery generation to support certain DQL operations, see “[DQL, DFC, and DFS queries](#)” on page 276.
- Configure search for wildcards, see “[Configuring a wildcard search](#)” on page 268.
- Route a query to a specific collection, see “[Routing a query to a specific collection](#)” on page 317.
- Turn on tracing for the Documentum query plugin, see “[Tracing Documentum queries](#)” on page 280.
- Customize facets and queries, see “[About Facets](#)” on page 333.

## 2.19 Administration APIs

The xPlore Admin API supports all xPlore administrative functions. The Admin API provides you with full control of xPlore and its components.



**Note:** Administration APIs are not officially supported. Contact the development team if you encounter issues using them.

Each API is described in the javadocs. Index service APIs are available in the interface `IFtAdminIndex` in the package `com.emc.documentum.core.fulltext.client.admin.api.interfaces`. This package is in the SDK jar file `dsearchadmin-api.jar`. System administration APIs are available in the interface `IFtAdminSystem` in the package `com.emc.documentum.core.fulltext.client.admin.api.interfaces` in the SDK jar file `dsearchadmin-api.jar`. Administration APIs are wrapped in a command-line interface tool (CLI). The syntax and CLIs are described in the chapter “[Automated Utilities \(CLI\)](#)”.

## 2.19.1 Open an admin connection

Create a façade implementation via FTAdminFactory. The following example uses the web service transport protocol to open a connection on the local xPlore instance. Parameters are String hostname or IP address, int port. String password, for example:

```
IFtAdminService srv = FTAdminFactory.getAdminService("127.0.0.1", 9300,  
"mypassword".toCharArray());
```

The password parameter is the xPlore administrator password.

## 2.19.2 Call an admin API

Invoke the admin API FTAdminFactory via the façade implementation in the package com.emc.documentum.core.fulltext.client.admin.api. The following example starts the primary instance:

```
IFtAdminService srv = FTAdminFactory.getAdminService("127.0.0.1", 9300,  
"mypassword".toCharArray());  
srv.startNode("primary");
```

## 2.19.3 Configuration APIs

Configuration APIs are available in the interface IFtAdminConfig in the package com.emc.documentum.core.fulltext.client.admin.api.interfaces. This package is in the SDK jar file dsearchadmin-api.jar.

### Engine configuration keys for setEngineConfig

- xhive-database-name: Managed by xDB.
- xhive-cache-pages: Number of pages to hold temporary data for all simultaneous xDB sessions. By default, xPlore sets the value to 25% of the maximum memory configured for the JVM.
- xhive-pagesize: Size in bytes of database pages, a power of 2. Usually matches filesystem pagesize.

### Indexing configuration keys for setIndexConfig

- index-requests-max-size: Maximum size of internal index queue
- index-requests-batch-size: Maximum number of index requests in a batch
- index-thread-wait-time: Maximum wait time in milliseconds to accumulate requests in a batch
- index-executor-queue-size: Maximum size of index executor queue before spawning a new worker thread

### Status update keys

- status-requests-max-size: Maximum size of internal status queue

- status-threadpool-core-size: Minimum number of threads used to process a single incoming request. Valid values: 1 - 100.
- status-threadpool-max-size: Number of threads used to process a single incoming request. Valid values: 1 - 100.
- status-executor-queue-size: Maximum size of index executor queue before spawning a new worker thread
- status-executor-retry-wait-time: Wait time in milliseconds after executor queue and worker thread maximums have been reached



**Note:** These parameters will not take effect by default, since the trackingDB and data libraries are updated in the same transaction. To change this, add the parameter *update-trackdb-in-same-transaction* with value *false* in index-config.

## Search configuration keys for setGlobalSearchConfig

- query-default-locale: Default locale for queries.
- query-default-result-batch-size: Default size of result batches. Default: 200.
- query-result-cache-size: Default size of results buffer. When this limit is reached, no more results are fetched from xDB until the client asks for more results. Default: 400.
- query-result-spool-location: Location to spool results. Default: <DOCUMENTUM\_HOME>/dss/spool
- query-default-timeout: Interval in milliseconds for a query to time out. Default: 60000. This setting is overridden by a setting in a client application: IDfxQuery TIMEOUT parameter, or the query\_timeout parameter in the dm\_ftengine\_config.
- query-thread-sync-interval: Interval after which results fetching is suspended when the result cache is full. For a value of 0, the thread waits indefinitely until space is available in the cache (freed up when the client application retrieves results). Default: 100 milliseconds.
- query-thread-max-idle-interval: Query thread is freed up for reuse after this interval. If the client application has not retrieved the result during this interval, the query times out. (Threads are freed immediately after a result is retrieved.) Default: 3600000 milliseconds.
- query-summary-default-highlighter: Class that determines summary. Default: com.emc.documentum.core.fulltext.indexserver.services.summary.DefaultSummary
- query-dynamic-summary-content-text-size: Content text size in bytes that is used to compute the dynamic summary. Default: 65536.
- query-summary-display-length: Size in bytes of summary to display. Default: 256.
- query-summary-highlight-begin-tag: customized tag to insert at beginning of summary. Default: none.

- `query-summary-highlight-end-tag`: customized tag to insert at end of summary. Default: none.
- `query-enable-dynamic-summary`: If context is not important, set to false. This returns as a summary the first n chars defined by the `query-summary-display-length` configuration parameter. No summary calculation is performed. For summaries evaluated in context, set to true (default).
- `query-index-covering-values`: The values specified in the path attribute are used for aggregate queries. These values are pulled from the index and not from the data pages.
- `query-facet-max-result-size`: Sets the maximum number of results used to compute facet values. For example, if `query-facet-max-result-size=12`, only 12 results are used to compute facets. If a query has many facets, the number of results per facet is reduced accordingly. Default: 10000.



# Chapter 3

## Managing Security

### 3.1 About security

xPlore does not have a security subsystem. Anyone with access to the xPlore host port can connect to it. You must secure the xPlore environment using network security components such as a firewall and restriction of network access. Secure the xPlore administrator port and open it only to specific client hosts. Passwords are encrypted with a FIPS PUB 140-2 compliant symmetric algorithm. Existing passwords are encrypted with DES. Documentum repository security is managed through individual and group permissions (ACLs). By default, security is applied to results before they are returned to the Documentum Server (native xPlore security), providing faster search results. xPlore security minimizes the result set that is returned to the Documentum Server. Documentum Server queues changes to ACLs and groups. The queue sometimes causes a delay between changes in the Documentum Server and propagation of security to xPlore. If the index agent has not yet processed a document for indexing or updated changes to a permission set, users cannot find the document. You can set up a separate index agent to handle changes to ACLs and groups. See “[Setting up index agents for ACLs and groups](#)” on page 80.

### 3.2 Changing search results security

For DQL queries, you can turn on Documentum Server security filtering to eliminate latency and support complete transactional consistency.



**Note:** When XQuery generation is turned off, search performance is worse. The following search enhancements do not work without XQuery: Facets, paging, and parallel summary.

To turn on security filtering in the Documentum Server for DQL queries:

1. Turn off XQuery generation. Add the following setting to dfc.properties on the DFC client application:

```
dfc.search.xquery.generation.enable=false
```

2. Open the iAPI tool from the Documentum Server Manager on the Documentum Server host or in Documentum Administrator.

3. To check your existing security mode, enter the following command:

```
retrieve,c,dm_ftengine_config  
get,c,l,ftsearch_security_mode
```

4. Enter the following command to turn off xPlore native security. Note lowercase L in the set and save commands:

```
retrieve,c,dm_ftengine_config
set,c,1,ftsearch_security_mode
0
save,c,1
reinit,c
```

5. Restart all xPlore instances.

#### *Using both security filters*

You can configure xPlore and the Documentum Server to use both security filters. This option applies only to DQL queries. With this option, results are not returned to the Documentum Server unless the user has permissions. After xPlore security is applied, the results are filtered again in the Documentum Server for changes to permissions that took place after the security update in xPlore.

Make sure that xPlore security is configured (default). Then use the following iAPI command:

```
retrieve,c,dm_ftengine_config
append,c,1,param_name
acl_check_db
append,c,1,param_value
T
save,c,1
```

### 3.3 Manually updating security

When you set the index agent UI to migration mode, ACLs and groups are indexed at the end of migration. ACL (*dm\_acl*) and group (*dm\_group*) objects are stored in XML format in the xPlore xDB. The XML for ACLs and groups is stored as a collection in xDB: *domain\_name/dsearch/ApplicationInfo/acl* or */ApplicationInfo/group*. The XML format is ACLXML for ACLs and GroupXML for groups. They are updated when a **Save**, **Save as new**, or **Destroy** event on an ACL or group takes place in the repository.



**Note:** The Security filter and permission set cache logic is designed to work with only one ACL and group collection. Do not create multiple ACL and group collections or change the default name of the collection.

The security filter is applied in xDB to filter search results per batch. The security filter receives the following information: User credentials, minimum permit level, whether MACL is enabled, privileges, and dynamic state of the user (group membership).

You can manually populate or update the ACL and group information in xPlore. A similar job in Documentum Server 6.7 and higher allows you to selectively replicate ACLs and groups. The script replicates all ACLs and groups. Use the job or script for the following use cases:

- You are testing Documentum indexing before migration.
- You use xPlore to index a repository that has no full-text system (no migration).
- Security in the index is out of sync with the repository from ftintegrity counts.



**Note:** To speed up security updates in the index, you can create a separate index agent for ACLs and groups. See “[Setting up index agents for ACLs and groups](#)” on page 80.

1. Locate the script `aclreplication_for_repositoryname.bat` or `.sh` in `xplore_home/setup/indexagent/tools`.
2. Edit the script before you run it. Locate the line beginning with “`%JAVA_HOME %\bin\java`” (Windows) or “`<$JAVA_HOME>/bin/java`”. Set the repository name, repository user, password, xPlore primary instance host, xPlore port, and xPlore domain (optional).

Check the console output for any errors or exceptions thrown. When you run the script, it prints the status of each object it tried to replicate. Alternatively, you can run the ACL replication job `dm_FTACLReplication` in Documentum Administrator. (Do not confuse this job with the `dm_ACLReplication` job.) By default, the job reports only the number of objects replicated. Setting the job argument `verbose` to true writes the status of each object in job report. You can selectively replicate only `dm_acl`, `dm_group`.



**Note:** The `dm_FTACLReplication` job is set to inactive after a successful run, meaning that this job cannot be scheduled to run periodically.

**Table 3-1: ACL replication job arguments**

Argument	Description
<code>-acl_where_clause</code>	DQL where clause to retrieve <code>dm_acl</code> objects.
<code>-group_where_clause</code>	DQL where clause to retrieve <code>dm_group</code> objects.
<code>-max_object_count</code>	Number of <code>dm_acl</code> and <code>dm_group</code> objects to be replicated. If not set, all objects are replicated.
<code>-replication_option</code>	Valid values: <code>dm_acl</code> , <code>dm_group</code> or both (default).
<code>-verbose</code>	Set to true to record replication status for each object in the job report. Default: false.

Argument	Description
<p>-standby_ftengine_id</p>	<p>HA active/active mode with multiple xPlore instances only:</p> <p>Specify an ID of a standby fulltext engine as the value of the argument to retrieve xPlore information from the dm_ftengine_config object where r_object_id is the specified ID.</p> <p>After you set this argument, you must restart the Documentum Java Method Server for the change to take effect.</p> <p> <b>Note:</b> A non-standby fulltext engine ID will cause the job to fail.</p>
<p>-ftEngineStandby</p>	<p>Dual mode (FAST and xPlore on two Documentum Servers) only: set this parameter to true.</p> <p><code>-ftEngineStandby T</code></p>

## 3.4 Changing the xDB administrator password

Perform the following steps to reset the xDB administrator or superuser password. All xPlore and index agent instances must share the same instance owner and password.



**Note:** If you change the Documentum repository owner password, you must also change it in the watchdog configuration file.

1. Reset the password from the xDB admin tool.
  - a. Navigate to `xplore_home/dsearch/xhive/admin`.
  - b. Run the script XHAdmin.bat (Windows) or XHAdmin (Linux).
  - c. Click the connection icon to log in. The password is the same as your xDB administrator password.
  - d. From the **Federation** menu, select **Change superuser password**. Enter the new and old passwords.
  - e. From the **Database** menu, select **Reset admin password**. Use the new superuser password that you created, and set the admin password. They can be the same.
2. Stop all xPlore instances.
3. Edit `indexserver-bootstrap.properties` of each instance. The file is located in `xplore_home/<wildfly_version>/server/DctmServer_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes`. Enter the `adminuser-password` and `superuser-password` that you created in the xDB admin tool (in plain text format).

4. Restart all xPlore instances. The passwords are now encrypted (FIPS-compliant encryption).
5. Change the WildFly password. Copy the new, encrypted admin user password from `indexserver-bootstrap.properties` to the following locations:
  - The file `dctm-users.properties` in `<xplore_home>/<wildfly_version>/server/DctmServer_PrimaryDsearch/configuration`.  
`admin=new_admin_password`
  - The file `PrimaryDsearch.properties` in `xplore_home/installinfo/instances/dsearch`:  
`ess.instance.password.encrypted=new_adm_password`

Repeat this step for all other xPlore instances, if any.

6. If you use CLI for backup and restore, edit the password in `xplore.properties`. This file is located in `xplore_home/dsearch/admin`. Copy the encrypted password from `indexserver-bootstrap.properties`.
7. The xDB Administrator password is reset. You can now use the new password to log into xPlore Administrator.

If you change the index agent installation owner password, in Windows you just have to change the services entry and on Linux you do not have to change anything. By default, xPlore accepts encrypted passwords, which means that you can log in to xPlore Administrator by copying the encrypted admin password from `indexserver-bootstrap.properties`.

To disable login with encrypted password, edit the file `indexserver-bootstrap.properties` located in `<xplore_home>/<wildfly_version>/server/DctmServer_PrimaryDsarch/deployments/dsearch.war/WEB-INF/classes` and add the following:

```
accept-encrypted-password=false
```

## 3.5 Configuring the security cache

Increase the cache size for large numbers of users or large numbers of groups that users can belong to. You can improve security filter performance by adjusting the following properties for security:

- `global-ace-cache-size`: Number of global ACL entries in the cache. Approximate maximum memory usage of ACE cache is  $(\text{global-ace-cache-size}) \times 120$  bytes
- `acl-cache-size`: Number of permissions calculated by ACL in the cache. Per-query LRU cache that contains ACLs and granted permissions for the user who is performing the search. Default: 100000.
- `global-acl-cache-user-count`: Number of users for global permission result calculated by ACL in the cache. Approximate maximum memory usage of ACL cache is  $(\text{acl-cache-size}) \times (\text{global-acl-cache-user-count}) \times 100$  bytes

- *enable-incremental-group-cache-update*: Enables incremental update of group cache. When its value is true, groups-in-cache-size will not take effect. Default: true.
- *groups-in-cache-size*: Number of users in the cache. The key is the user name, the value is a list of groups the user belongs to. Global LRU cache that is shared between search sessions. Default: 1000.
- *batch-size*: Size of batches sent for search results security filtering. Default: 800.
- *owner-group-cache-size*: Number of entries in the owner group cache. Default: 100000.

1. Stop all xPlore instances
2. Edit indexserverconfig.xml. For information on viewing and updating this file, see ["Modifying indexserverconfig.xml" on page 54](#).
3. Adjust values in the *security-filter-class* element as in this example:

```
<security-filter-class default="true" class name="com.emc.documentum.core.fulltext.indexserver.services.security.SecurityJoin" name="documentum">
  <properties>
    <property value="100000" name="owner-group-cache-size"/>
    <property value="1000000" name="acl-cache-size"/>
    <property value="20" name="global-acl-cache-user-count"/>
    <property value="2000000" name="global-ace-cache-size"/>
    <property value="true" name="enable-incremental-group-cache-update"/>
    <property value="800" name="batch-size"/>
  </properties>
</security-filter-class>
```

4. If necessary, change the Groups-in cache cleanup interval by adding a property to the *security-filter-class* properties. The default is 7200 sec (2 hours).
5. Validate your changes using the validation tool described in ["Modifying indexserverconfig.xml" on page 54](#).

```
<property name="groupcache-clean-interval" value="7200" />
```

## 3.6 Troubleshooting security

### Viewing security in the log

Check dsearch.log using xPlore administrator. Choose an instance and click Logging. Click dsearch log to view the following information:

- The XQuery expression. For example, search for the term *default*:

```
QueryID=PrimaryDsearch$f3087f7a-fb55-496a-bf0a-50fb1e688fa1,
query-locale=en,query-string=declare option xhive:fts-analyzer-class
'com.emc.documentum.core.fulltext.indexserver.core.index.xhive.IndexServerAnalyzer';

declare option xhive:ignore-empty-fulltext-clauses 'true';
declare option xhive:index-paths-values "dmftmetadata
//owner_name,dmftsecurity/acl_name,dmftsecurity/acl_domain";
let $libs := collection('/TechPubsGlobal/dsearch/Data')
let $results := for $dm_doc score $
in $libs/dmftdoc[(dmftmetadata//a_is_hidden = "false") and
(dmftversions/iscurrent = "true") and
(. ftcontains "test" with stemming using stop words default)]
```

```

order by $s descending return $dm_doc return (for $dm_doc in subsequence($results,
1,351) return <r>
{for $attr in $dm_doc/dmftmetadata//*[local-name()='
object_name','r_modify_date','r_object_id','r_object_type','
r_lock_owner','owner_name','r_link_cnt','r_is_virtual_doc',
'r_content_size','a_content_type','i_is_reference','r_assembled_from_id','r_has_frzn_
assembly','a_compound_architecture','i_is_replica','r_policy_id','subject','title')]
return <attr name='{local-name($attr)}' type='{$attr/@dmfttype}'>{string($attr)}</
attr>}{xhive:highlight(($dm_doc/dmftcontents/dmftcontent/
dmftcontentref,$dm_doc/dmftcustom))<attr name='score'
type='dmdouble'>{string(dsearch:get-score($dm_doc))}
</attr></r>) is running

```

- Security filter applied and security statistics. For example:

```

2016-04-14 15:39:37,811 INFO [Search-Thread-4]
c.e.d.c.f.i.services.security.SecurityJoin - {DYNAMIC_GROUPS=[],
Total-values-from-data-page=1, Group-cache-fill-time=1,
QueryID=PrimaryDsearch$5a07fd73-d828-4478-9b12-6788b5221749,
Total-res-with-no-dmftdoc=0, Total-owner-is-group-cache-hits=0,
Total-matching-group-probes=0, Total-values-from-index-keys=4,
Security-evaluation-time=100, Total-owner-not-group-cache-hits=0,
Filter-input=2200, Filter-output=200, Total-GLOBAL-ACE-cache-hits=0, Total-owner-
group-probes=0, HANDLE_ACL_CACHE_TIME=0, Security-double-check-input=0,
Total-groups-in-cache-hits=0, Total-matching-owner-group-probes=0,
MACL_ENABLED=true, USE_GLOBAL_ACE_CACHE=true,
HANDLE_OWNER_TIME=0, Total-ACL-index-probes=0,
USE_GLOBAL_ACL_CACHE=true, GET_DOC_INFO_TIME=0,
Minimum-permit-level=2, Total-Group-index-probes=1,
Total-ACL-cache-hits=0, HANDLE_ACL_INDEX_TIME=0,
Owner-group-cache-fill-time=0}

```

In the example, the query returned 2200 hits to filter. 2000 were filtered out, returning 200 results to the client application.

When INFO is enabled for security package, the following information is saved in dsearch.log:

- Minimum-permit-level. Returns the minimum permit level for results for the user. Levels: 0 = null | 1 = none | 2 = browse | 3 = read | 4 = relate | 5 = version | 6 = write | 7 = delete
- Filter-output: Total number of hits after security has filtered the results.
- Total-values-from-index-keys: Number of index hits on owner\_name, acl\_name and acl\_domain for the document.
- QueryID: Generated by xPlore to uniquely identify the query
- Total-values-from-data-page: Number of hits on owner\_name, acl\_name and acl\_domain for the document retrieved from the data page.
- Filter-input: Number of results returned before security filtering.
- Total-group-index-probes: Keeps track of how many group indexes are probed. After cache is populated for a user, if the same user executes another query, this value should be zero.
- Total-matching-group-probes: How many times the query added a group to the group-in cache.
- Total-ACL-index-probes: How many times the query added an ACL to the cache. If this value is high, you can speed up queries by increasing the ACL cache size.

- Total-groups-in-cache-hits: Number of times the group-in cache contained a hit.
- Total-ACL-cache-hits: Number of times the ACL cache contained a hit.
- Group-cache-fill-time: How much time in milliseconds group caching was completed by retrieving and iterating indexes from xDB or by running xQuery.
- Security-evaluation-time: How long in milliseconds it took to complete the whole security evaluation.
- Total-owner-is-group-cache-hits: Number of hits for the ownerIsGroup cache.
- Total-owner-not-group-cache-hits: Number of hits for the ownerIsNotGroup cache.
- Total-owner-group-probes: Number of times xquery is run.
- Owner-group-cache-fill-time: Time required to run xquery and fill in the cache in milliseconds.
- Total-matching-owner-group-probes: Number of times the owner group query returns *yes*.
- Total-res-with-no-dmftdoc: Number of documents that are not dmftdoc.
- Total-GLOBAL-ACE-cache-hits: Number of hits for global ACE cache.
- Security-double-check-input: Number of documents before security double check.
- MACL\_ENABLED: If MACL is enabled.
- USE\_GLOBAL\_ACE\_CACHE: If global ACE cache is used.
- USE\_GLOBAL\_ACL\_CACHE: If global ACL cache is used.
- GET\_DOC\_INFO\_TIME: time spent getting document owner name and acl information in milliseconds.
- HANDLE\_OWNER\_TIME: time spent checking if the user is document owner in milliseconds.
- HANDLE\_ACL\_CACHE\_TIME: Time spent getting permission from ACL cache.
- HANDLE\_ACL\_INDEX\_TIME: time spent calculating permission from ACL index.

## Verifying security settings in the Documentum Server

Use iAPI to verify that *dm\_fulltext\_index\_user* is registered to receive events for security updates (changes to *dm\_acl* and *dm\_group*) with the following commands. They return at least one ACL object ID and one group object ID:

```
? ,c,select r_object_id from dm_type where name='dm_acl'  
? ,c,select r_object_id from dm_type where name='dm_group'
```

Verify that the ACL IDs are registered for the events *dm\_save*, *dm\_destroy*, and *dm\_saveasnew*. Verify that the group IDs are registered for the events *dm\_save* and *dm\_destroy*, for example:

```
? ,c,select registered_id,event from dmi_registry where user_name='dm_fulltext_index_user'
```



**Note:** If HA is configured, ensure all dm\_fulltext\_index\_user\* are registered.



## Chapter 4

# Managing the Documentum Index Agent

## 4.1 About the Documentum index agent

The xPlore index agent is a multithreaded Java application running in the application server. The index agent processes index queue items generated by Documentum Server, applies filters, and prepares SysObjects for indexing.



**Note:** Documentum Administrator reports all queue items, including those that are subsequently filtered out by the index agent.

The xPlore installer includes the index agent and its configurator.

A dm\_ftindex\_agent\_config object represents the index agent in normal mode. This object is configured by the index agent configurator. For more information about the index agent config object, refer to the *Documentum Server System Object Reference Guide*.

### Documentum Server objects created by index agent

When you configure an index agent, it creates the following objects in the Documentum Server:

Object	Attributes
dm_ftengine_config	For a full list of attributes, see “ <a href="#">dm_ftengine_config</a> ” on page 397.
dm_acl	object_name: dm_fulltext_admin_acl; owner_name: Name of user specified at installation acl_class: 3 accessor_permit: 7, 7, 3 accessor_name: dm_owner, dm_fulltext_admin, dm_world
dm_fulltext_index	index_name is_standby: Indicates whether the index agent is in use or in standby mode. install_loc: Type of agent (dsearch or fast) ft_engine_id: Specifies the associated dm_ftengine_config object.

Object	Attributes
dm_ftindex_agent_config	index_name queue_user: Identifies the full-text user who is registered in dmi_registry for full-text events.

## Index agent loading

In both migration and normal mode, index agent configuration is loaded from indexagent.xml in the index agent deploy directory. In normal mode, the configuration is also read from the dm\_ftindex\_agent\_config object. If there is a conflict, the settings in the config object override the settings in indexagent.xml.

### 4.1.1 Documentum attributes that control indexing

The *a\_full\_text* attribute is defined for the dm\_sysobject type. The *a\_full\_text* attribute is set to true whenever a sysobject is created. All sysobject subtypes inherit this Boolean attribute that controls whether the content files of an object are indexed. Indexing is performed whenever a Save, Saveasnew, Checkin, Destroy, Branch, or Prune operation is performed on the object. Users with Sysadmin or Superuser privileges can change the *a\_full\_text* setting.

Properties of the format object determine which formats are indexable. If the value of the *can\_index* property is set to true, the content file is indexable. 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, you can ensure indexing by creating a rendition in an indexable format. Use Documentum Content Transformation Services or third-party client applications to create the rendition. For a full list of supported formats, see Oracle Outside In documentation.

Some formats are not represented in the repository by a format object. Only the properties of objects in that format are indexed. The formats.csv 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 has no format object, create a format object in the repository and map the supported mime\_type to the format in formats.csv.

Documents are selected for indexing in the Documentum Server based on the following criteria:

- If *a\_full\_text* attribute is false, the content is not indexed. Metadata is indexed.
- If *a\_full\_text* attribute is true, content is indexed based on the *can\_index* and *format\_class* attributes on the *dm\_format* associated with the document:
  1. If an object has multiple renditions and none of the renditions have a *format\_class* value of *ft\_always* or *ft\_preferred*, each rendition is examined starting with the primary rendition. The first rendition for which *can\_index* is true is indexed, and no other renditions are indexed.

2. If an object has a rendition whose format\_class value is ft\_preferred, each ft\_preferred rendition is examined in turn starting with the primary rendition. The first ft\_preferred rendition that is found is indexed, and no other renditions are indexed.
3. If an object has renditions with a format\_class value of ft\_always, those renditions are always indexed.



**Note:** Index agent filters can override the settings of a\_full\_text and can\_index. See “Configuring Index Agent filters” on page 85.

Sample DQL to determine these attribute values for the format *bmp*:

```
? ,c,select can_index, format_class from dm_format where name = 'bmp'
```

To find all formats that are indexed, use the following command from iAPI:

```
? ,c,select name,can_index from dm_format
```

The *dm\_ftengine\_config* object has a repeating attribute *ft\_collection\_id* that references a collection object of the type *dm\_fulltext\_collection*. Each ID points to a *dm\_fulltext\_collection* object. It is reserved for use by Documentum Server client applications.

## Indexing aspect attributes

Properties associated with aspects are not indexed by default. If you wish to index them, issue an ALTER ASPECT statement to identify the aspects you want indexed.

**Table 4-1: Syntax for full-text indexing of aspects**

Syntax	Description
FULLTEXT SUPPORT ADD ALL	Defines all properties of the aspect for indexing.
FULLTEXT SUPPORT ADD property_list	Defines for indexing only those aspect properties listed in property_list.
FULLTEXT SUPPORT DROP ALL	Stops indexing of all properties of the aspect.
FULLTEXT SUPPORT DROP property_list	Stops indexing of those aspect properties listed in property_list.

When you add or drop indexing for aspect properties, clean the DFC BOF cache for the changes to take effect.

1. Stop the index agent.
2. On the index agent host, delete the directory for the DFC bof cache. The directory is set by dfc.data.dir in dfc.properties. For example:
 

```
<xplore_home>\wildfly_version\server\DtcmServer_Indexagent\data\Indexagent\cache\<documentum_server_version>\bof\<repository_name>
```
3. Start the index agent.

Only new objects are affected. The index is not updated to add or drop aspect property values for aspects attached to existing objects.

### **Indexing lightweight sysobjects**

Lightweight objects inherit the attributes of the parent as shared, not private, attributes. Inheritance allows many lightweight objects to share the same attributes and reduce the storage requirement for some of the private content-related attributes like ID and size. These attributes are computed on demand. DQL queries cannot access the shared parent attribute values.

The fulltext\_support attribute on the object type is used for lightweight sysobjects (LWSOs) to determine how they are indexed. Valid values:

Value	Description
0	No support
1	Light support
2	Full support

## **4.2 Starting the index agent**

To start or stop the index agent service, run the following commands. The script names contain the index agent server name that you specified when you configured the index agent. Default: Indexagent.

1. Start the instance.

- Windows

The Documentum Index\_agent Windows service, or *indexagent\_home*  
\\<wildfly\_version>\\server\\startIndexagent.cmd

- Linux

*indexagent\_home*/<wildfly\_version>/server/startIndexagent.sh

2. Start the index agent UI. Use your browser to start the index agent servlet.

`http://host:port/IndexAgent/login_dss.jsp`

Every index agent URL has the same URL ending: IndexAgent/login\_dss.jsp.  
Only the port and host differ.

- *host* is the DNS name of the machine on which you installed the index agent.
- *port* is the index agent port number that you specified during configuration (default: 9200).

3. In the login page, enter the user name and password for a valid repository user and optional xPlore domain name.

4. Choose one of the following:

- **Start Index Agent in Normal Mode:** The index agent will index content that is added or modified after you start.
- **Start new reindexing operation:** All content in the repository is indexed (migration mode) or reindexed. Filters and custom routing are applied. Proceed to the next step in this task.
- **Continue:** If you had started to index this repository but had stopped, start indexing. The date and time you stopped is displayed.

#### *Viewing index agent details*

Start the index agent and click **Details**. You see accumulated statistics since last index agent restart and objects in the indexing queue. To refresh statistics, return to the previous screen and click **Refresh**, then view **Details** again.

## 4.3 Silent index agent startup

You can start or shut down the index agent through the index agent web application. You can also script the start in normal mode or shutdown using Documentum Server, iAPI or DFC. Starting in migration mode cannot be scripted.

### Setting startup in Documentum Server

Set the `start_index_agents` parameter in `server.ini` to `T`. At Documentum Server startup, the Server checks whether the index agent associated with the repository has started in normal mode. If not, and `start_index_agents` is `T`, the Server starts the index agent in normal mode using the `dm_FTIndexAgentBoot` job.



**Note:** The `dm_FTIndexAgentBoot` job uses the Index Agent Servlet and starts the index agent only if the WildFly instance has started.

When the index agent is configured in SSL mode and `start_index_agents = T` is set in `server.ini`, you must perform the following tasks to make sure that the index agent starts up along with the repository startup.

- Documentum Server must be run in SSL mode to make SSL handshake successful between Documentum Server and the wildfly instance of the index agent.
- When anonymous SSL (default one, without any certificates) is used, an anonymous cipher (such as, `TLS_DH_anon_WITH_AES_128_CBC_SHA` for `AES-128`) must be configured in `stand-alone.xml` of IndexAgent's wildfly instance.

### Silent startup and shutdown with iAPI

Use the `retrieve` and `dump` commands to get the `index_name` attribute of the `dm_fulltext_index` object. You use this attribute value in the start or stop script. For example:

```
API> retrieve,c,dm_fulltext_index
...
3b0004d280000100
API> dump,c,1
...
USER ATTRIBUTES

index_name : Repo_ftindex_01
...
```

Now use the retrieve and dump commands to get the object\_name attribute of the dm\_ftindex\_agent\_config object. You use this attribute value in the start or stop script. For example:

```
retrieve,c,dm_ftindex_agent_config
...
0800277e80000e42
API> dump,c,1
...
USER ATTRIBUTES
object_name : Config13668VMO_9200_IndexAgent
```

Use the *apply* command to start or stop (shutdown) the index agent, and to view its current status. Syntax:

```
apply,c,,FTINDEX_AGENT_ADMIN,NAME,S,<index_name of dm_fulltext_index>,
AGENT_INSTANCE_NAME,S,<object_name of dm_ftindex_agent_config>,ACTION,
S,start|shutdown|status
```

The following example starts one index agent:

```
apply,c,NULL,FTINDEX_AGENT_ADMIN,NAME,S,LH1_ftindex_01,AGENT_INSTANCE_NAME,
S,Config13668VMO_9200_IndexAgent,ACTION,S,start
```

To start or stop all index agents, replace the index agent name with *all*. For example:

```
apply,c,NULL,FTINDEX_AGENT_ADMIN,NAME,S,LH1_ftindex_01,
AGENT_INSTANCE_NAME,S,all,ACTION,S,shutdown
```

Follow with these commands to get the results:

```
API> next,c,qNumber
...
OK
API> dump,c,qNumber
```

Where <Number> is the number of execution times that starts at 0 for the first command execution and increments by 1 with each execution.

Viewing the current index agent status returns one of the following:

- 0: The index agent is running.
- 100: The index agent has been shut down.
- 200: The index agent has a problem.

## Setting startup with a list of file IDs

You can script startup to index a list of documents like the list generated by ftintegrity.

1. Start the index agent servlet in normal mode.
2. Create a text file with the object ID. Save as `ids.txt` in the `WEB-INF/classes` directory of `xplore_home/<wildfly_version>/server/DctmServer_IndexAgent/deployments/IndexAgent.war/`. (Specify the actual path to your index agent web application.)

The objects in `ids.txt` are automatically submitted for indexing. You can select the refeed task for these objects from the list of refeed tasks displayed from the index agent UI, and then export the task details to check whether the objects have been successfully indexed.



**Note:** After the refeed task has been created, the `ids.txt` will be removed from the `WEB-INF/classes` directory of `xplore_home/<wildfly_version>/server/DctmServer_IndexAgent/deployments/IndexAgent.war`.

## Startup from the Java command line

Use the following command:

```
java com.documentum.server.impl.utils.IndexAgentCtrl -docbase_name
repositoryName -user_name userName -action actionPerformed
-index_agent instanceName
```

where `-action` argument value is one of the following: `start` | `shutdown` | `status` | `reset`.



**Note:** If you specify the instance name of IndexAgent, the specified action only takes effect on the specified IndexAgent; if no instance name of IndexAgent is specified, the specified action takes effect on all instances.

## Silent startup and shutdown using DFC

The following method gets the `dm_fulltext_index` object, the `index_name` attribute, and sets the DQL query:

```
public void shutdownIA(IDfSession sess) throws DfException
{
    IDfPersistentObject FTIndexObj = (
        IDfPersistentObject) sess.getObjectByQualification(
        "dm_fulltext_index where is_standby = false");

    String indexName = FTIndexObj.getString("index_name");

    //Query definition
    String query = "NULL,FTINDEX_AGENT_ADMIN,NAME,S," +
        indexName + ",AGENT_INSTANCE_NAME,S,all,ACTION,S,shutdown";

    DfClientX clientX = new DfClientX();
    IDfQuery q = clientX.getQuery();

    q.setDQL(query);

    try
    {
        IDfCollection col = q.execute(sess, IDfQuery.DF_APPLY);
    }
    catch (DfException e)
    {
```

```
    e.printStackTrace();
}
```

For startup, replace *shutdown* with *start* in the query definition.

## 4.4 Manipulating an Index Agent with DM\_TICKET

If you use an index agent servlet to start, stop, or check the status of an index agent instance, you can use *DM\_TICKET* instead of the user password in the servlet URL. For example, the servlet URL for starting the index agent instance looks similar to:

```
https://hostname:port/IndexAgent/servlet/IndexAgent?command=start
&docbase=docbase_name&user=username&ticket=DM_TICKET&instance=index_agent_instance_name
```



### Notes

- The user must have SysAdmin privileges or higher.
- The index agent instance name must be found in the *indexagent.xml* configuration file or the *dm\_ftindex\_agent\_config* object.

## 4.5 Setting up index agents for ACLs and groups

By default, you configure an index agent for each Documentum repository that is indexed. You can also set up multiple index agents to index *dm\_acl* and *dm\_group* separately from *sysobjects*.

1. Create a second index agent. Run the index agent configurator and give the agent instance a name and port that are different from the first agent. The configurator is the file *configIndexagent.bat* or *configIndexagent.sh* in *indexagent\_home/setup/indexagent*.
2. Edit *indexagent.xml* for the second index agent. This file is located in *indexagent\_home/<wildfly\_version>/server/DctmServer\_Indexagent2/deployments/IndexAgent.war/WEB-INF/classes*.
3. Add one parameter set to your new *indexagent.xml* file. Set the value of *parameter\_name* to *index\_type\_mode*, and set the value of *parameter\_value* to *aclgroup* as follows:

```
<indexer_plugin_config>
  <generic_indexer>
    <class_name>... </class_name>
    <parameter_list>
      ...
      <parameter>
        <parameter_name>index_type_mode</parameter_name>
        <parameter_value>aclgroup</parameter_value>
      </parameter>
    </parameter_list>
  </generic_indexer>
</indexer_plugin_config>
```

4. In the *indexagent.xml* for *sysobjects* (the original index agent), add a similar parameter set. Set the value of *parameter\_name* to *index\_type\_mode*, and set the value of *parameter\_value* to *sysobject*.

5. Restart both index agents. (Use the scripts in `indexagent_home/<wildfly_version>/server` or the Windows services.)

## 4.6 Configuring the index agent after installation

The Documentum index agent is configured for the first time through the index agent configurator, which can be run after installing xPlore. For information on the configurator, see *Documentum xPlore Installation Guide*.

Parameter default values have been optimized for most environments. They can be changed later using iAPI or by editing `indexagent.xml`, which is located in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`. For descriptions of the settings, see “[Index agent configuration parameters](#)” on page 399.

### Limit content size for indexing

You can set a maximum size for content that is indexed. You set the actual document size, not the size of the text within the content. To set the maximum document size, edit the `contentSizeLimit` parameter within the parent element `exporter`. The value is in bytes. Default: 20 MB.



**Note:** You can also limit the size of text within a file by configuring the CPS instance. Set `max_text_threshold` in bytes (default: 10 MB).

### Exclude ACL and group attributes from indexing

By default, all attributes of ACLs and groups are indexed. You can specify that certain attributes of ACLs and groups are not indexed. Add an `acl_attributes_exclude_list` and `group_attributes_exclude_list` element to the parent element `indexer_plugin_config/generic_indexer/parameter_list`.

### Change the local content storage location

When you configure the index agent, you select a local content temporary staging location. You can change this location by editing the `local_content_area` element in `indexagent.xml`. This file is located in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`. Restart the index agent web application after editing this file.



**Note:** For multi-instance xPlore, the temporary staging area for the index agent must be accessible from all xPlore instances.

### Specify the DFC cache directory

By default, the DFC cache data is stored in the following directory:

`<xplore_home>/<wildfly_version>/server/DctmServer_Indexagent/data/Indexagent`

You can change this directory by specifying the value of the `<dfc.data.dir>` property in the `dfc.properties` file.

`dfc.properties` location: `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`.



**Note:** On Linux, path names are case-sensitive. `Indexagent` and `IndexAgent` are two different directories.

## Prevent resubmitting of the same object for indexing

Errors occur when an object is submitted for indexing multiple times resulting in duplicate object IDs being processed.

To prevent the xPlore client from resubmitting an object for indexing when it has already been submitted but not yet indexed:

1. Manually create a new file `dsearchclient.properties` in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`.
2. In the file, add the following property:  
`disable.xploreclient.timeout=true`
3. Restart the index agent and the feature is turned on. The xPlore client will not resubmit the same object for indexing until the previously submitted one has been processed.

### 4.6.1 Overview of Index Agent filters

Index Agent filters enable you to control which objects are indexed by the xPlore search engine.

The Index Agent provides default filters that enable you to control which objects of type `SysObject`, and its sub-types, are indexed. For more information, see [Default Index Agent filters](#).

You can also create your own custom filters as explained in [Create custom filters](#).

#### Default Index Agent filters

The xPlore Index Agent provides the following default filters:

- **Type:** Allows you to define filters based on the `r_object_type` property.
- **Cabinet:** Allows you to define filters based on the `i_cabinet_id` property.
- **Folder:** Allows you to define filters based on the `i_folder_id` property.

The default filters allow you to create rules (called filter configurations) that define whether objects are indexed. You can create, apply, and manage the filter configurations using the Index Agent UI.

 **Notes**

- Some object types, cabinets, and folders are included or excluded by default to improve performance.
- From version 16.4, the Index Agent loads the default filters from its class path instead of the Documentum Server repository. If there are old default filters in the Documentum Server repository, the Index Agent ignores them.

While the default filters are stored with the Index Agent, the filter configurations (rules) are saved in the Documentum Server repository. The Index Agents load the filter configurations from the repository and refresh their cache at the configured interval. For information about configuring the cache refresh interval, see [Configure interval to refresh filter cache](#).

## Default filter modes

There are two modes in which a default filter can operate. Depending on your requirements, you can select either of the following modes:

- *Include Mode*: Allows you to define rules that specify which objects will be included in the index. By default, all objects that are not explicitly included in the filter are excluded. This mode is also useful if the types of objects that you want to include are fewer.
- *Exclude Mode*: Allows you to define rules that specify which objects will be excluded from the index. By default, all objects not explicitly excluded in the filters are included. This mode is also useful if the types of objects that you want to exclude are fewer.

For example, you might want to index all other types of objects except Java library files (JAR files). You can accomplish this by selecting the *Exclude Mode* for the **Type** filter and adding the type `dmc_java_library` to the **Types to Exclude** filter list.



**Note:** You must use the Include mode for the Cabinet or Folder default filters carefully. No object is indexed if the filter lists for these filters are empty in the Include mode.

## How default filters work

The Index Agent processes SysObject objects against all default filters. If it finds a rule that excludes an object, the object is filtered out. If none of the rules exclude the object, the object is indexed.



**Note:** An object can be filtered out if it has an object property that matches any item in an Exclude mode filter list or if it has an object property that does not match any of the items in an Include mode filter list.

The following table lists the filter configurations and their outcomes for some common scenarios:

**Table 4-2: Common default filter configurations and their outcomes**

<b>Filter Configurations</b>	<b>Outcome</b>
All modes are Exclude and all Exclude filter lists (Exclude Type/Exclude Cabinet/Exclude Folder) are empty.	All objects are indexed.
All modes are Include and all Include filter lists (Include Type/Include Cabinet/Include Folder) are empty.	All objects are filtered.
Type mode is Exclude. Cabinet mode is Exclude. Folder mode is Include. The type of the object is not in the Exclude list, cabinet not in the Exclude list, folder is in the Include list.	The object is indexed.
Type mode is Exclude. Cabinet mode is Exclude. Folder mode is Include. The type of the object is not in the Exclude list, cabinet not in the Exclude list, folder not in the Include list.	The object is filtered.

## Create custom filters

If required, you can also implement your own custom filters and deploy them along with the default filters. You can use custom filters to filter content based on any object attribute, such as creation date.

You need to complete the following tasks to create and implement a custom filter:

- Develop a custom filter.
- Deploy your custom filter as a Business Object Framework (BOF) module.
- Restart the Index Agent to implement the new filter.

A custom filter should implement the Documentum Foundation Classes (DFC) interface `com.documentum.fc.indexagent.IDfCustomIndexFilter`. It should return one of the following `DfCustomIndexFilterAction` actions for the content that you are filtering:

- INDEX: To index the content and metadata.
- SKIP: To skip indexing of the content and metadata.
- METADATA: To index only the metadata of the content.

For information on creating a BOF filter, see [Injecting data and supporting joins](#).

### TBO filtering of indexable objects

```
public class NoteFTFilter extends DfSingleDocbaseModule implements
IDfCustomIndexFilter
{
    public DfCustomIndexFilterAction getCustomIndexAction (IDfPersistentObject
```

```

object) throws DfException
{
    //We don't filter any non-sysobjects such as Groups and ACLs
    if (!(object instanceof IDfSysObject))
        return DfCustomIndexFilterAction.INDEX;
    String objectType = object.getString("r_object_type");
    if ("dm_note".equals(objectType))
    {
        return DfCustomIndexFilterAction.SKIP;
    }
    return DfCustomIndexFilterAction.INDEX;
}
}

```

## Configure interval to refresh filter cache

Changes to the filter configuration in the Index Agent UI are saved to the Documentum Server repository immediately. However, the different Index Agents connected to the repository download the filter configurations and refresh their respective cache only after a configured interval because of performance considerations. Therefore, updates to the filter configuration in the Index Agent UI are applied only after an interval.

The reload and refresh interval for the Index Agent cache is configured in the file `indexagent.xml`, which is located in the following folder:

```
xplore_home/wildfly_version/server/DctmServer_Indexagent/deployments/
IndexAgent.war/WEB-INF/classes
```

This file contains an XML element `update_filter_cache_interval` (within the parent element `indexagent_instance`) which defines the interval in seconds. The default value is 1800 seconds.

### 4.6.2 Configuring Index Agent filters

You can view, add, modify, and delete filters using the Index Agent UI. For information about how filters work, see [Overview of Index Agent Filters](#).

1. Click the **Filters** tab to create or update filters.
2. From the **Choose Filters Mode** section, select the required mode for the three default filters.



**Note:** By default, the **Exclude Mode** is selected for all the default filters.

Based on the mode you select; the respective filters are displayed. For example, if you select the **Include Mode** for the **Type** filter, the **Types to Include** filters are displayed.

3. Click **Save**.
4. To add one or more filters, type one or more values separated with a comma and click **Add**. For example:

- To exclude one or more object types: 1) ensure the **Exclude Mode** is selected for **Type**; 2) under **Types to Exclude** filter, type the object types you want to exclude; and 3) click **Add**.



**Note:** Adding a type does not add its subtypes.

- To include one or more folders: 1) ensure the **Include Mode** is selected for **Folder**; 2) under **Folders to Include** filter, type the folder paths that you want to include; and 3) click **Add**.

5. To remove a filter, click the cross icon under the **Remove** column, which corresponds to the filter.



**Note:** The changes to the filter list are immediately saved when you add or remove a filter.

The new filters are applied only on the documents that have not been indexed as yet. They do not impact the documents already indexed or filtered out. For information about monitoring an index agent and viewing filters as they are being applied, see [Monitor a running Index Agent](#).

### 4.6.3 Monitor a running Index Agent

The **CS Status** tab displays information about the status of an Index Agent and helps you monitor a running Index Agent. If the Index Agent is not running, the tab displays static information about the Index Agent configuration.

#### Event Register Status

The **Event Register Status** section displays which Documentum Server events trigger xPlore Index Agents to index. The events listed in this section are categorized by the underlying object type for which the events are generated.

- **Queue User Name:** The user name that is used for Index Agent to register the events in the docbase. For example, `dm_fulltext_index_user`.
- **Type Name:** The object type for which events are registered by the queue user. For example, `dm_group`, `dm_acl`, `dm_sysobject` are the object types that are indexed by the Index Agent by default.
- **Event:** The event types that are registered for the given object type.

#### DM Queue Item Status

This section displays the status of the actual events (`dmi_queue_item` objects) that were generated for the Index Agent. The status includes—the number of events that were either not handled or failed during handling—by the Index Agents connected to the repository.

- **Item State(task\_state):** The status of the queue item. For example:

- Empty: An empty field indicates that the event/queue item was “not handled.”
- failed: A failed status indicates that the event was “handled but failed.”

 **Note:** This section may display other states depending on the Index Agent configuration.

- **Sign Off User(sign\_off\_user):** The instance name of the Index Agent that handled the relevant queue items.
- **Count:** The number of queue items with the given state/status.

 **Note:** This information is loaded from the repository directly when you open the **CS Status** tab.

## Runtime Filters

If the Index Agent is started, this section displays the following details about the Index Agent filters that are running:

- The interval at which the filter cache is refreshed.
- The Types (objects types) that have been excluded or included (depending on which Type mode is chosen.)
- The Cabinet names and IDs that have been excluded or included (depending on which cabinet mode is chosen.)
- The Folder paths and IDs that have been excluded or included (depending on which folder mode is chosen.)
- The Filter classes that have been applied (the three default filters and custom filters, if any).

If the Index Agent is not started, the following information is displayed in the **Runtime Filters** section:

- Type names that will be excluded or included (depending on which type mode is chosen)
- Cabinet names that will be excluded or included (depending on which cabinet mode is chosen)
- Folder paths that will be excluded or included (depending on which folder mode is chosen)

 **Note:** When the Index Agent is not started, the values displayed are same as the values displayed on the **Filters** page. Other values are not shown because they are not loaded by the Index Agent.

#### 4.6.4 Sharing content storage

In the indexing process, the Index Agent prepares the indexing file for the CPS, then CPS reads the file and the extracts tokens.

There are two methods the Index Agent can prepare the indexing files, that are similar to the IAPI commands of Documentum Server:

- `getfile`: Download the file from the Documentum Server to a local folder.
- `getpath`: Do not download the file, but read the actual file path in Documentum Server.

By default, the Index Agent retrieves content from the content storage area to the index agent temporary content location (a `getfile` operation). This temporary file is deleted after it has been indexed.

To optimize performance, you can share the content storage (a `getpath` operation). With shared content storage, CPS has direct read access to the content. No content is streamed. You must map the path to the file store in the index agent web application.

The Index Agent parameters are as follows:

- `local_content_area`: IndexAgent temporary content location for `getfile` operation.
- `all_filestores_local`: When this parameter is true, the IndexAgent performs the `getpath` operation on the indexing objects.
- `local_filestore_map`: It only takes effect when the `all_filestores_local` parameter is false.
  - `local_filestore`
    - `store_name`: Matches the attribute of `a_storage_type` of an indexing object. Refer *Documentum Server Administration and Configuration Guide* for more information.
    - `local_mount`: The local mounted folder for CPS to access the file in Documentum Server.

The indexing file path that the Index Agent has prepared is added as a value of the `dmftcontentref` parameter in `dmftxml`, which is the request body that the Index Agent sends to the Index Server. The Index Server sends this request to CPS for token extraction.

The following table describes how these parameters work together to affect the final indexing file path (the value of `dmftcontentref`):

<b>all_filestores_local</b>	<b>local_filestore_map</b>	<b>local_filestore_map.local_filestore matches indexing object's a_storage_type</b>	<b>similar IAPI command</b>	<b>dmftcontentref value in dmftxml sent to Index Server</b>
FALSE	empty	-	getfile	The file path under local_content_area.
	not empty	NO	getfile	The file path under local_content_area.
	not empty	YES	getpath	The file path under local_filestore.local_mount.
TRUE	empty	-	getpath	The direct file path as getpath in Documentum Server
	not empty	NO		
	not empty	YES		

The debug log on the Index Server Admin Console is displayed if the parameters are applied.



**Note:** The content storage area must be unencrypted and mountable as read-only by the Index Agent and xPlore hosts.

1. On the index agent host, open `indexagent.xml`, located in `indexagent_home/wildfly_version/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`. If multiple index agents are installed on this host, an integer is appended to the IndexAgent WAR file name, for example, `IndexAgent1.war`.
2. Set the parameters in the exporter element:
  - If the file system paths to content are the same on the Documentum Server host and xPlore host, change the value of the child element, `all_filestores_local` to true.

For example, check all the file store names and their locations from the Document Server IAPI command line:

```
API> ?,c,select fs.name, lt.file_system_path from dm_filestore fs, dm_location lt where
fs.root=lt.object_name
name          file_system_path
-----
filestore_01      C:\Documentum\data\repo\content_storage_01
thumbnail_store_01  C:\Documentum\data\repo\thumbnail_storage_01
streaming_store_01 C:\Documentum\data\repo\streaming_storage_01
```

If the Index Agent and the CPS can access all the file store paths which will be indexed, `all_filestores_local` can be set as true.

- If the file system paths are different, but the Index Agent and CPS can access the Documentum Server files through a mounted folder, set the `all_filestores_local` as false, and add a file store map within the exporter element. Specify the store name and a local mapping folder for each file store.

For example, find all `store_name` in Documentum Server IAPI command line:

```
API> ?,c,select r_object_id, name from dm_filestore
r_object_id      name
-----
28190bc480000100 filestore_01
28190bc480000101 thumbnail_store_01
28190bc480000102 streaming_store_01
Take filestore_01 for example, and check its path on Documentum Server:
API> ?,c,select file_system_path from dm_location where object_name = (select
root from dm_filestore where name = 'filestore_01')
file_system_path
-----
C:\Documentum\data\repo\content_storage_01
```

If the C drive located on Documentum Server is mapped to the Z drive on CPS server, then the CPS can access all the files of `filestore_01` through the path `Z:\Documentum\data\repo\content_storage_01`.

The following configuration can be added to the exporter element of `indexagent.xml` file:

```
<all_filestores_local>false</all_filestores_local>
<local_filestore_map>
  <store_name>filestore_01</store_name>
  <local_mount>Z:\Documentum\data\repo\content_storage_01</local_mount>
  </local_filestore>
</local_filestore_map>
```

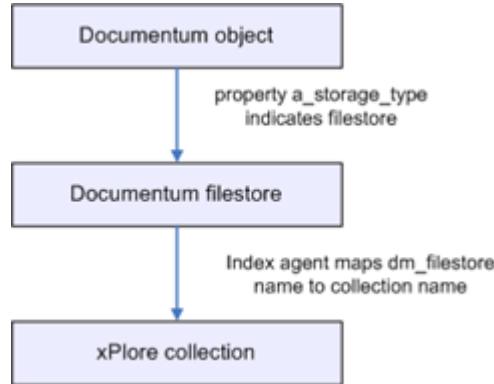


**Note:** Update the `file_system_path` attribute of the `dm_location` object in the repository to match this `local_mount` value, and then restart the Documentum Server

3. Save the `indexagent.xml` and restart the index agent instance.

#### 4.6.5 Mapping Server storage areas to collections

A Documentum Server file store can map to an xPlore collection. File store mapping to a collection allows you to keep collection indexes separate for faster ingestion and retrieval.



**Figure 4-1: Filestore mapping to an xPlore collection**

You can create multiple full-text collections for a repository for the following purposes:

- Partition data
  - Scale indexes for performance
  - Support storage-based routing
1. Open `indexagent.xml`, located in the indexing agent WAR file in the directory (`<xplore_home>/<wildfly_version>/server/DctmServer_IndexAgent/deployments/IndexAgent.war/WEB-INF/classes`).
  2. Add `partition-config` and its child elements to the element `indexer` to map file stores to collections.

In the following example, filestore\_01 maps to collection 'coll01', and 02 to 'coll02'. The rest of the repository is mapped to the default collection. Each repository has one default collection named `default`.

```

<partition_config>
  <default_partition>
    <collection_name>default</collection_name>
  </default_partition>
  <partition>
    <storage_name>filestore_01</storage_name>
    <collection_name>coll01</collection_name>
  </partition>
  <partition>
    <storage_name>filestore_02</storage_name>
    <collection_name>coll02</collection_name>
  </partition>
</partition_config>
  
```

## 4.7 Migrating documents

### 4.7.1 Migrating content (reindexing)



**Note:** You cannot filter content after it has been indexed. For information on removing documents from the index, see “[Removing entries from the index](#)” on page 106.

1. Start the index agent and log in to the UI.
2. Choose **Start new reindexing operation**.

This indexes all indexable documents in the repository except for the documents excluded by the index agent filter. Updates are performed for existing documents. Custom routing is applied. To skip custom routing for documents that are already indexed, edit indexserverconfig.xml. Set *allow-move-existing-documents* to false in index-config.

### 4.7.2 Migrating documents by object type

To migrate or reindex many documents of a specific object type, do not use the DQL or object ID file in the index agent UI. Instead, perform the following steps in migration mode, which allows a restart of the indexing process. Index agent filters are applied in this migration.

1. Edit `indexagent.xml`, located in `indexagent_home/<wildfly_version>/server/DctmServer_indexagent/deployments/IndexAgent.war/WEB-INF/classes`.
2. Add a `parameter_list` element with the following content to the `indexagent_instance` element. Substitute the object type name for the value of the `parameter_value` element. For example:

```
<parameter_list>
  <parameter>
    <parameter_name>type_based_migration</parameter_name>
    <parameter_value>TYPE_NAME_HERE</parameter_value>
  </parameter>
</parameter_list>
```



**Note:** The `parameter_list` element can contain only one `parameter` element.

3. Stop and restart the index agent using the scripts in `indexagent_home/<wildfly_version>/server` or using the Windows services panel.
4. Log in to the index agent UI and choose **Start new reindexing operation**.
5. When indexing has completed (on the **Details** page, no more documents in the queue), click **Stop IA**.
6. Run the acreplication script to update permissions for users and groups in xPlore. See “[Manually updating security](#)” on page 64.
7. Update the `indexagent.xml` file to index another type or change the `parameter_value` to `dm_document`.

### 4.7.3 Migrating a limited set of documents

If you want to migrate few object types, you can use the index agent UI. To migrate a large set of objects by type, see “[Migrating documents by object type](#)” on page 92.

1. Replicate ACLs and groups to xPlore by running the acrreplication script. See “[Manually updating security](#)” on page 64.
2. Start the index agent in normal mode.
3. Select **DQL** and select the type.
4. Repeat for each type that you want indexed.
5. Navigate to the Refeed Tasks page to display the Refeed Tasks List. You can view the refeed task status, pause, resume or delete tasks, and export failed ID lists or an xml file with error messages.



**Tip:** Use the **Select All Tasks** and **Deselect All Tasks** options to select and deselect all the tasks respectively in the list.

To remove from the index documents that have already been indexed, see “[Removing entries from the index](#)” on page 106.

### 4.7.4 Moving collection documents without reindexing

To move documents from one collection to another without reindexing:

1. Choose one of the following options to edit either the `indexagent.xml` or a domain element in `indexserverconfig.xml`:
  - From `indexer_plugin_config/generic_indexer/parameter_list`, add the following code to `indexagent.xml`, change the `target_collection` (can be a list of collection names, separated by comma), and restart the index agent in normal mode:

```
<parameter>
  <parameter_name>collection</parameter_name>
  <parameter_value>target_collection</parameter_value>
</parameter>
```

- Add the following code to a domain element in `indexserverconfig.xml`, change the `target_collection` (target collection in the rule element can be a list of collection names, separated by comma), and restart xPlore:

```
<backend-collection-routing class-name="FtCollectionRoutingOnXPath"
document-category="dftxml" default-collection="target_collection">
  <properties>
    <property value="false" name="honor-tracking-info"/>
  </properties>
  <collection-routing>
    <rule collection="target_collection" condition="true()" />
  </collection-routing>
</backend-collection-routing>
```

2. Edit `migrateCollection.groovy`, and then go to %XPORE%/dsearch/admin and run `xplore.bat/sh -f scripts/migrateCollection.groovy`.

The source collection is now in update\_and\_search state and can be updated from the Admin Console.



**Note:** Only data usage collection is supported for target and source collections.

## 4.8 Using ftintegrity

*ftintegrity* and the state of index job (in Documentum Server 6.7 or higher) are used to verify indexing after migration or normal indexing. The utility verifies all types that are registered in the *dmi\_registry\_table* with the user *dm\_fulltext\_index\_user*. The utility compares the object ID and *i\_vstamp* between the repository and xPlore. You can compare metadata values, which compares object IDs and the specified attributes.

Run *ftintegrity* as the same administrator user who started the instance.



**Note:** *ftintegrity* can be very slow, because it performs a full scan of the index and content. Do not run *ftintegrity* when an index agent is migrating documents.

Run the *ftintegrity* index verification tool after migration or restoring a federation, domain, or collection. The tool is a standalone Java program that checks index integrity against repository documents. It verifies all types that are registered to *dmi\_registry\_table* with the user *dm\_fulltext\_index\_user*, comparing the object ID and *i\_vstamp* between the repository and xPlore.

Use the option *-checkType* to check a specific object type. Use the option *-checkMetadata* to check specific single-value attributes (requires *-checkType*).

1. Navigate to *xplore\_home/setup/indexagent/tools*.
2. Open the script *ftintegrity\_for\_repositoryname.bat* (Windows) or *ftintegrity\_for\_repositoryname.sh* (Linux) and edit the script. Substitute the repository instance owner password in the script (replace *<password>* with your password). The tool automatically resolves all parameters except for the password.
3. Optional: Add the option *-checkfile* to the script. The value of this parameter is the full path to a file that contains sysobject IDs, one on each line. This option compares the *i\_vstamp* on the ACL and any groups in the ACL that is attached to each object in a specified list. If this option is used with the option *-checkUnmaterializeLWSO*, *-CheckType*, *-StartDate*, or *-EndDate*, these latter options are not executed.

For example:

```
....FTStateOfIndex DSS_LH1 Administrator mypassword  
Config8518VMO 9300 -checkfile ...
```

4. Optional: Add the option *-checkType* to compare a specific type in the Documentum Server and index. You can run the script for one type at a time.

The tool checks sysobject types or subtypes. It does not check custom types that are not subtypes of dm\_sysobject.

For example:

```
$JAVA_HOME/bin/java ... -checkType dm_document
```

5. Optional: Add the option -checkMetadata at the end of the script. This argument requires a path to a metadata.txt file that contains a list of required single-valued (not repeating) metadata fields to check, one attribute name per line. (Create this file if it does not exist.) This option applies only to a specific type.

For example, add the following to the ftintegrity script in *xplore\_home/setup/indexagent/tools*:

```
$JAVA_HOME/bin/java ... -checkType dm_document  
-checkMetadata C:/xplore/setup/indexagent/tools/metadata.txt
```

Metadata mismatches are recorded in several files in *xPlore\_home/setup/indexagent/tools*:

- Object-Metadata-mismatch.txt: contains all the objects with metadata that has inconsistencies.
- Object-Metadata-match.txt: contains all the objects with metadata that has valid consistencies.
- Object-fetched-from-cs.txt: contains all the object IDs that are fetched from the Documentum Server.
- Object-fetched-from-xPlore.txt: contains all the object IDs that are fetched from xPlore.
- ObjectId-indexOnly.txt: contains all the object IDs that exist only in xPlore.

The ftintegrity tool generates the following files:

- ObjectId-common-version-mismatch.txt: contains ACLs that are out of sync as the content of the elements acl\_name|domain/acl i\_vstamp in docbase/acl i\_vstamp in xDB.
- ObjectId-common-version-match.txt: contains all the object IDs with consistent versions.
- ObjectId-dctmOnly.txt: contains groups that are out of sync as the content of the elements Mismatching i\_vstamp group:/Sysobject ID: id/Group ids in dctm only:/group id.
- ObjectId-indexOnly.txt: contains all the object IDs that exist only in xPlore.



**Note:** All optional arguments must be appended to the end of the java command line.

### 4.8.1 ftintegrity output

Output from the script is like the following:

```
Executing stateofindex

Connected to the docbase SAMPLEDOCBASE
2011/03/14 15:41:58:069 Default network framework: http
2011/03/14 15:41:58:163 Session Locale:en
2011/03/14 15:41:59:913 fetched 1270 object from docbase for type dm_acl
2011/03/14 15:41:59:913 fetched 1270 objects from xPlore for type dm_acl
2011/03/14 15:42:08:428 fetched 30945 object from docbase for type dm_sysobject
2011/03/14 15:42:08:428 fetched 30798 objects from xPlore for type dm_sysobject
2011/03/14 15:42:08:756 fetched 347 object from docbase for type dm_group
2011/03/14 15:42:08:756 fetched 347 objects from xPlore for type dm_group
2011/03/14 15:42:09:194 **** Total objects from docbase : 32215 ****
2011/03/14 15:42:09:194 **** Total objects from xPlore : 32068 ****
2011/03/14 15:42:09:194 17 objects with different ivstamp in DCTM and Index Server
2011/03/14 15:42:09:194 147 objects in DCTM only
2011/03/14 15:42:09:194 0 objects in Index Server only

ftintegrity is completed.
```

Interpreting the output:

- objects from dm\_acl and dm\_group: Numbers fetched from repository (docbase) and xPlore.
- match ivstamp: Objects that have been synchronized between Documentum Server and xPlore.
- different ivstamp: Objects that have been updated in Documentum Server but not yet updated in the index.
- objects in DCTM only: These objects are in the repository but not xPlore for one or more of the following reasons:
  - Objects failed indexing.
  - New objects not yet indexed.
  - Objects filtered out by index agent filters.
- objects in Index Server only: Any objects here indicate objects that were deleted from the repository but the updates have not yet propagated to the index.

In the example, the ACLs and groups totals were identical in the repository and xPlore, so security is updated. There are 147 objects in the repository that are not in the xPlore index. They were filtered out by index agent filters, or they are objects in the index agent queue that have not yet been indexed. To eliminate filtered objects from the repository count, add the *usefilter* argument to ftintegrity (slows performance).

## 4.8.2 ftintegrity result files

The script generates four results files in the tools directory:

- ObjectId-common-version-match.txt: This file contains the object IDs and *i\_vstamp* values of all objects in the index and the repository and having identical *i\_vstamp* values in both places.
- ObjectId-common-version-mismatch.txt: This file records all objects in the index and the repository with identical object IDs but nonmatching *i\_vstamp* values. For each object, it records the object ID, *i\_vstamp* value in the repository, and *i\_vstamp* value in the index. The mismatch is on objects that were modified during or after migration. You can resubmit this list after you start the index agent in normal mode. Click **Object File** and browse to the file.
- ObjectId-dctmOnly.txt: This report contains the object IDs and *i\_vstamp* values of objects in the repository but not in the index. This file does not report on inconsistent metadata checks using the *-checkMetadata* option. The objects in this report could be documents that failed indexing, documents that were filtered out, or new objects generated in the repository during or after migration. You can resubmit this list after you start the index agent in normal mode. Click **Object File** and browse to the file. To check whether filters were applied during migration, run the following DQL query. If one or more rows are returned, a filter was applied.

```
? ,c,select r_object_id,object_name,primary_class from dmc_module where any  
a_interfaces='com.documentum.fc.indexagent.IDfCustomIndexFilter'
```

- ObjectId-indexOnly.txt

This report contains the object IDs and *i\_vstamp* values of objects in the index but not in the repository.

These objects were removed from the repository during or after migration, before the event has updated the index.

You can input the ObjectId-common-version-mismatch.txt file into the index agent UI to see errors for those files. After you have started the index agent, select **Object file**. Browse to import the file and then click **Submit**. Navigate to the Refeed Tasks page to display the Refeed Tasks List. You can view the refeed task status, pause, resume or delete tasks, and export failed ID lists or an xml file with error messages.

### 4.8.3 Running the state of index job

Repository configuration for Documentum Server 6.7 SPx or later installs a job called *state of index* (object\_name: dm\_FTStateOfIndex, subject: State of Index). Patches for previous versions of Documentum Server also install the job. This job is implemented as a Java method and runs in the Documentum Server Java method server.

You can also use *ftintegrity* tool to check the consistency between the repository and the xPlore index. The *ftintegrity* script calls the *dm\_FTStateOfIndex* job.



**Note:** *ftintegrity* and the *dm\_FTStateOfIndex* job can be very slow, because they perform a full scan of the index and content. Do not run *ftintegrity* or the *dm\_FTStateOfIndex* job when an index agent is migrating documents.

The *state of index* job compares the index content with the repository content. Execute the *state of index* job from Documentum Administrator (DA). The job generates reports that provide the following information:

- Index completeness and comparison of document version stamps.
- Status of the index server: Disk space usage, instance statistics, and process status.
- Total number of objects: Content correctly indexed, content that had some failure during indexing, and objects with no content

To disable the job, view the job properties in Documentum Administrator and change the state to *inactive*.

### 4.8.4 state of index and ftintegrity arguments

The *state of index* and *ftintegrity* arguments are similar, with some slight differences. You can set the *state of index* job argument values in DA using the *ftintegrity* form of the argument. Job arguments are case sensitive. *ftintegrity* arguments are not.

**Table 4-3: State of index and ftintegrity arguments**

Job arg	ftintegrity option	Description
-batchsize	batchsize (argument, not option)	Number of objects to be retrieved from the index in each batch. The default value is 10000.

Job arg	ftintegrity option	Description
-check_file	-CheckFile	<p>The value of this parameter is the full path to a file that contains sysobject IDs, one on each line. This option compares compares the i_vstamp on the ACL and any groups in the ACL that is attached to each object in a specified list. If this option is used with the option -checkUnmaterializeLWSO, -CheckType, -StartDate, or -EndDate, these latter options will not be executed.</p> <p>ACLs that are out of sync will be listed in ObjectID-common-version-mismatch.txt as the content of the elements acl_name domain/acl i_vstamp in docbase/acl i_vstamp in xDB. Groups that are out of sync will be listed in ObjectID-dctmOnly.txt as the content of the elements Mismatching i_vstamp group:/Sysobject ID: id/Group ids in dctm only:/group id...</p>
-check_type	-checkType	Specifies a specific object type to check (includes subtypes). Must be a subtype of dm_sysobject (not dm_acl or dm_group). Other types will not be checked.
-check_metadata	-checkMetadata	Specifies path to a file metadata.txt in <i>xplore_home/setup/indexagent/tools</i> . This file contains one single-valued attribute name per line. Requires a type specified in the -check_type argument.
-check_unmaterialized_lwso	-checkLWSO	Sets whether to check unmaterialized lightweight sysobjects during comparison. Default: false.

Job arg	ftintegrity option	Description
-collection_name	Not available	<p>Compares index for the specified collection to data in the repository.</p> <p>Default: All collections. Cannot use this argument with the <i>ftintegrity</i> script.</p>
-end_date	-EndDate	<p>Local end date of sysobject <i>r_modify_date</i>, for range comparison. Format: MM/dd/yyyy HH:mm:ss</p>
-ftengine_standby	-ftEngineStandby	<p>Dual mode (FAST and xPlore on two Documentum Servers) only: set the parameter -ftEngineStandby to true:</p> <p>-ftEngineStandby T</p> <p>After you set this argument, you must restart the Documentum Java Method Server for the change to take effect.</p>
-standby_ftengine_id <ID>	-standbyFtEngineId <ID>	<p>HA active/active mode with multiple xPlore instances only:</p> <p>Specify an ID of a standby fulltext engine as the value of the argument to retrieve xPlore information from the dm_ftengine_config object where r_object_id is the specified ID. The corresponding dm_fulltext_index must have is_standby set to true.</p> <p>After you set this argument, you must restart the Documentum Java Method Server for the change to take effect.</p> <p> <b>Note:</b> A non-standby fulltext engine ID will cause the job to fail.</p>

<b>Job arg</b>	<b>ftintegrity option</b>	<b>Description</b>
-fulltext_user	-fulltextUser	Name of user who owns the xPlore instance. For dual mode (FAST and xPlore), the user is <i>dm_fulltext_index_user_01</i> .
-get_id_in_indexing	Not available	This setting is deprecated. If specified, IDs that have not yet been indexed will be dumped to a file, ObjectId-in-indexing.txt. Default: False. Cannot use this argument with the <i>ftintegrity</i> script.
-sort_order	-sortOrder	Sets the sort order of object IDs. Valid values: NLS_SORT values in Oracle 11g. For hex values, use BINARY. This parameter should be set to BINARY whenever user's Oracle NLS_LANGUAGE is not marked as English.
-start_date	-StartDate	Local start date of sysobject <i>r_modify_date</i> , for range comparison. Can be used to evaluate the time period since the last backup. Format: MM/dd/yyyy HH:mm:ss
-timeout_in_minute	-timeout	Number of minutes to time out the session. Default: 1.
<i>-usefilter value</i>	-usefilter	Evaluates results using the configured index agent filters. Default: false. The job runs more slowly with -usefilter. For information on filters, see <a href="#">"Configuring Index Agent filters" on page 85</a> .
-verbose	-verbose	Set to true to get more information. Default: false. For ftintegrity script, add <i>-verbose T</i> at the end of java command.

Job arg	ftintegrity option	Description
NA	-blackout	<p>Specifies a list of blackout hours separated by comma. Valid values are integers 0 thru 23. <i>ftintegrity</i> sleeps during the blackout hours.</p> <p>Example: 8,9,10,11,12  <i>ftintegrity</i> sleeps from 8:00 to 12:59 everyday.</p>
NA	-exportMismatch	<p>Specifies the location where <i>ftintegrity</i> generates the <i>ids.txt</i> file. <i>ftintegrity</i> exports the mismatch object IDs to <i>ids.txt</i>. All objects listed in this file are automatically submitted for re-indexing. Index Agent removes <i>ids.txt</i> when all listed objects are submitted.</p> <p>You must set this option to:  <i>&lt;xplore_home&gt;/wildfly_version/server/&lt;DctmServer_IndexAgent&gt;/deployments/IndexAgent.war/WEB-INF/classes</i></p>
NA	-exportMaxIDNumber	<p>Specifies the maximum number of mismatch objects in <i>ids.txt</i> to re-index in one batch. If this option is not specified, all mismatch objects in <i>ids.txt</i> are submitted for re-indexing at once.</p>
NA	-exportCollisionWaitMinutes	<p>Specifies the maximum amount of time in minutes for <i>ftintegrity</i> to wait in case <i>ids.txt</i> already exists. If an existing <i>ids.txt</i> is processed and removed by Index Agent before the specified amount of time elapses, <i>ftintegrity</i> exports new object IDs right after the removal of the existing <i>ids.txt</i>. If the existing <i>ids.txt</i> takes longer time to process than the specified waiting time, the exporting of new object IDs is cancelled.</p>

Job arg	ftintegrity option	Description
NA	-deleteObjectsOnlyInxPlore	Determines whether or not to delete the objects listed in <code>ObjectId-indexOnly.txt</code> from xPlore.  Valid values: T and F  Default value: F

In addition, the job is installed with the -queueperson and -windowinterval arguments set. The -queueperson and -windowinterval arguments are standard arguments for administration jobs and are explained in the *OpenText Documentum Server Administration and Configuration Guide*.

## 4.9 Indexing documents in normal mode

In normal mode, indexable documents are queued up for indexing. You can also index a small set of documents in the normal mode page of the index agent.

1. Start the index agent and click **Start Index Agent in normal mode**.
2. You can select objects for indexing with a DQL statement or a list of objects. Use *ftintegrity* or the *state of index* job to generate the list. Edit the output file `ObjectId-common-version-mismatch.txt` to remove all data except object IDs.



**Note:** Do not use this option for many documents. Instead, use migration mode (reindexing), which allows a restart of the indexing process. Migration also reindexes all ACLs and groups, keeping security up to date.

3. Navigate to the Refeed Tasks page to display the Refeed Tasks List. You can view the refeed task status, pause, resume or delete tasks, and export failed ID lists or an xml file with error messages.

## 4.10 Resubmitting documents for indexing

To resubmit documents for indexing, you can submit a DQL or a file of object IDs from the Index Agent UI. You can also copy a file of object IDs to the file system of the IndexAgent server to automatically submit the objects for indexing.



### Notes

- To get a list of objects that failed indexing, you need to run *ftintegrity* or the *state of index* Documentum Server job to get a list of objects that failed in indexing. See “[Using ftintegrity](#)” on page 94. Remove all data from the file `ObjectId-common-version-mismatch.txt` except object IDs.

- To copy an object ID list to the IndexAgent server, you need to create a text file with the object ID. Save as `ids.txt` in the `WEB-INF/classes` directory of `xplore_home/<wildfly_version>/server/DctmServer_IndexAgent/deployments/IndexAgent.war/`. (Specify the actual path to your index agent web application.) For more details see the *Setting startup with a list of file IDs* topic in “[Silent index agent startup](#)” on page 77 of the *OpenText Documentum xPlore Installation Guide*.

To submit a DQL or file of object IDs from the Index Agent UI, do the following:

1. Start the index agent in normal mode. You will see a page that allows you to input a selected list of objects for indexing.
2. Select one of these options:
  - Click **DQL** and select the type. You can include all versions and add a clause if necessary. To preview the DQL result, click **Preview**.
  - Click **Object File** and browse to import a file.
3. Click **Submit**.
4. Navigate to the Refeed Tasks page to display the Refeed Tasks List. You can view the refeed task status, pause, resume or delete tasks, and export failed ID lists or an xml file with error messages.

Completed tasks older than 60 days are deleted.



**Tip:** Use the **Select All Tasks** and **Deselect All Tasks** options to select and deselect all the tasks respectively in the list.

## 4.11 Indexing auto retry

When indexing fails for some documents, you need to run the `ftintegrity` utility to verify the indexing and get a list of objects that failed indexing so as to resubmit them for indexing. This process can be very slow because the `ftintegrity` performs a full scan of the index and content for comparison. With the indexing auto retry feature, the index agent automatically resubmits documents that failed indexing for self-recovery and logs indexing errors into the xPlore metrics database.

A retry indexing queue item is generated for an item that failed indexing, but if the item fails indexing again on the retry, no new queue item will be generated and no error message will be logged for the item that failed the indexing retry. Highly automated and reliable, the indexing auto retry feature minimizes the need to run the time-consuming `ftintegrity` utility as well as manual administrative work to refeed documents that failed indexing. During the indexing auto retry process for a document in the Documentum Server that failed indexing:

1. The index agent creates a new retry queue item in the repository for the document that failed indexing.
2. The index agent sends an error message to xPlore and xPlore saves it in the metrics database.

3. The retry queue item will be picked up by the normal indexing process.

You can view summary and detailed error and warning messages raised by document indexing failures by running the following reports under **Diagnostic and Utilities**:

- IA Message Summary Per Hour: Displays the number of error messages and warning messages raised by document indexing failures during each hour on a specified date.
- IA Message Summary Per Day: Displays the number of error messages and warning messages raised by document indexing failures during each day in a specified month.
- IA Message Summary Per Month: Displays the number of error messages and warning messages raised by document indexing failures during each month of a specified year.
- IA Message Record: Displays detailed error and/or warning messages raised by document indexing failures during a specified period of time.

### 4.11.1 Disabling indexing auto retry

The indexing auto retry feature is enabled by default. To disable this feature, edit `indexagent.xml` in `<xplore_home>/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes` and add the following parameter under the `<indexagent_instance>` element:

```
<indexagent_instance>
  ...
  <parameter_list>
    <parameter>
      <parameter_name>disable_retry</parameter_name>
      <parameter_value>true</parameter_value>
    </parameter>
  </parameter_list >
  ...
</indexagent_instance>
```

After you configure this setting, restart the index agent for the change to take effect:

After logging in to the Index Agent Administration UI (default: `http://host:9200/IndexAgent`), click **Stop IA**; then select **Start Index Agent in Normal Mode** and click **Submit**.

### 4.11.2 Enable indexing auto retry for documents indexed with warnings

By default, retry indexing queue items are generated for item that failed indexing with errors rather than warnings, which normally mean metadata was indexed but not content. Warnings can be permanent or transient. Enabling indexing auto retry for items indexed with warnings refeed items indexed with transient warnings, such as when content is cleaned up by the index agent before fetched by CPS. To enable indexing auto retry for documents indexed with warnings, edit `indexagent.xml` in `<xplore_home>/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes` and add the following parameter under the `<indexagent_instance>` element:

```
<indexagent_instance>
  ...
  <parameter_list>
    <parameter>
      <parameter_name>include_warning_for_retry</parameter_name>
      <parameter_value>true</parameter_value>
    </parameter>
  </parameter_list >
  ...
</indexagent_instance>
```

After you configure this setting, restart the index agent for the change to take effect:

After logging in to the Index Agent Administration UI (default: `http://host:9200/IndexAgent/login_dss.jsp`), click **Stop IA**; then select **Start Index Agent in Normal Mode** and click **Submit**.

## 4.12 Removing entries from the index

You can remove certain object types, or objects that meet other criteria such as dates, from the index. You can execute a DQL query to get object IDs of the documents that you wish to delete from the index. Save the list of object IDs in a text file.

1. Navigate to `xplore_home/dsearch/xhive/admin`.
2. Open `deletedocs.properties` in a text editor.
3. Make sure that the host and port values correspond to your environment.
4. Set the value of `dss_domain` to the xPlore domain from which you wish to delete indexed documents.
5. If you want to delete indexed documents not in the default collection and not of the default category (dftxml); for example, collection acl and category acl, uncomment the following lines and specify the collection and category from which to delete indexed documents.  
`#dss_collection=default#dss_category=dftxml`
6. Change the value of the key `file_contains_id_to_delete` to the path to your object IDs. Alternatively, you can list the object IDs, separated by commas, as the value of the key `ids_to_delete`.

7. On Windows, run `deleteDocs.bat`; on Linux, run `deleteDocs.sh`.

The deletedocs utility records activity in a log in `xplore_home/dsearch/xhive/admin/logs/deleteDocs.log`.

## 4.13 Indexing metadata only

Use iAPI to set the `can_index` attribute of a `dm_format` object to False. As a result, contents of that format are not full-text indexed. For example:

```
retrieve,c,dm_format where name = 'tiff'  
set,c,1,can_index  
F  
save,c,1
```

## 4.14 Making types non-indexable

1. In Documentum Administrator, select the object type (**Administration > Types** in the left pane).
2. Right-click for the list of attributes.
3. Under the **Info** tab, clear the **Register for indexing** option to exclude this type from indexing.

This setting turns off indexing events. If the option is selected but you have enabled an index agent filter, the filter overrides this setting.

## 4.15 Making metadata non-searchable

You can make specific metadata non-searchable by adding a subpath definition. For example, to make the Documentum attribute `r_full_content_size` non-searchable, create a subpath like the following. Set the full-text-search attribute to false.

```
<sub-path path="dmftmetadata//r_full_content_size" type="double"  
full-text-search="false" value-comparison="true" returning-contents="true" />
```

If you make this change after indexing, reindex objects to make the metadata non-searchable.

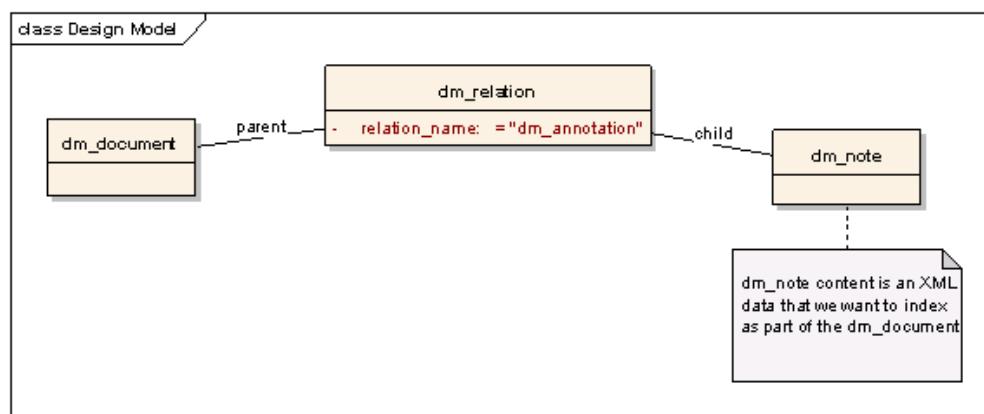
Documentum object types can be marked as non-indexed in Documentum Administrator. See “[Making types non-indexable](#)” on page 107.

## 4.16 Injecting data and supporting joins

In xPlore indexing, the metadata is added to the dmftmetadata node of dftxml. Content is added to the dmftcontent node. You can enhance document indexing with metadata or content from outside a Documentum repository. To inject metadata or content, create a type-based object (TBO) in DFC and deploy it to the repository. When you create a TBO for your type and deploy the TBO, DFC invokes your customization for objects of the custom type. For information on creating TBOs, refer to *OpenText Documentum Foundation Classes Development Guide*.

To support queries on multiple related objects (joins), create a type-based object (TBO) in DFC that denormalizes multiple objects into a single XML structure. The denormalized data is placed under the dmftcustom node of dftxml. Related content such as an MS Word document or a PDF is referenced under dmftcontent. Define specific indexes for data that is added to these nodes.

The following pseudocode example adds data from an attached object into the dftxml for the main object. The example triggers reindexing of the main object when the attached object is updated. It also prevents the attached object from being indexed as a standalone object. The following figure shows the object model. The customization adds the data for the dm\_note object to the dm\_document object.



**Figure 4-2: Custom indexing object model**

### Sample TBO class

Extend DfPersistentObject and override the method customExportForIndexing as shown in the following example.

```

public class AnnotationAspect extends DfSysObject
{protected void customExportForIndexing (IDfFtExportContext context)
throws DfException
{
  super.customExportForIndexing(context);
  Document document = context.getDocument(); //gets the main document

  NodeList nodes = document.getElementsByTagName( DfFtXmlElementNames.ROOT );
  
```

```

Element rootElem = nodes.getLength() > 0 ? (Element) nodes.item(
0) : null;
if (rootElem == null)
{
    rootElem = document.createElement( DfFtXmlElementNames.ROOT );
    document.appendChild( rootElem );
}

nodes = rootElem.getElementsByTagName( DfFtXmlElementNames.CUSTOM );
Element dmftcustom = nodes.getLength() > 0 ? (Element) nodes.item(
0) : null;
if (dmftcustom == null)
{
    dmftcustom = document.createElement( DfFtXmlElementNames.CUSTOM );
    rootElem.appendChild( dmftcustom );
}

Element mediaAnnotations = document.createElement("mediaAnnotations");
dmftcustom.appendChild(mediaAnnotations);

DocumentBuilder domBuilder = null;
try
{
    domBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
}
catch (ParserConfigurationException e)
{
    throw new DfException(e);
}
IDfCollection childRelations = getChildRelatives("dm_annotation");
while (childRelations.next())
{
    Element annotationNode = document.createElement("annotation");
    mediaAnnotations.appendChild(annotationNode);
    try
    {
        IDfId id = childRelations.getTypedObject().getId("child_id");
        // This will get the dm_note object
        IDfDocument note = (IDfDocument) getSession().getObject(id);
        ByteInputStream xmlContent = note.getContent();
        Document doc = domBuilder.parse(xmlContent);

        // Add the node content
        annotationNode.appendChild(doc);

        // Add a node for the author of a note
        Element authorElement = document.createElement("author");
        authorElement.setTextContent(note.getString("r_modifier"));
        annotationNode.appendChild(authorElement)
    }
    catch (SAXException e)
    {
        // Log the error
    }
    catch (IOException e)
    {
        // Log the error
    }
}
childRelations.close();}}
```

### Generated dftxml

```

<dmftdoc>
...
<dmftcustom>
    <mediaAnnotations>
        <annotation>
            <content>
                This is my first note
            
```

```
</content>
<author>Marc</author>
</annotation>
<annotation>
  <content>
    This is my second note
  </content>
  <author>Marc</author>
</annotation>
</mediaAnnotations>
</dmftcustom>
</dmftdoc>
```

## Triggering document reindexing

In the previous example, when the related object is modified, the main object is not reindexed. The following TBO for the related type (dm\_note) triggers reindexing of the dm\_document object. The TBO applies to dm\_note objects. When the object is saved after modification, the object is queued for indexing by calling IDfDocument.queue. Substitute the index agent name in the queue method. If you have more than one agent, call the queue method for each one.

```
public class NoteTbo extends DfDocument implements IDfBusinessObject
{protected synchronized void doSaveEx (boolean keepLock, String versionLabels,
Object[] extendedArgs) throws DfException
{
  super.doSaveEx(keepLock, versionLabels, extendedArgs);
  IDfCollection parentRelations = getParentRelatives("dm_annotation");
  while (parentRelations.next())
  {
    IDfId id = parentRelations.getTypedObject().getId("parent_id");
    IDfDocument annotatedObject = (IDfDocument) getSession().getObject(id);
    annotatedObject.queue("dm_fulltext_index_user", "
dm_force_ftindex", 1, false, new DfTime(), "");}}
```

## Preventing indexing of attached objects

If the attached object does not need to be searchable, you can prevent indexing.

## Indexing the injected metadata

In addition to your TBO, set up an index for your injected metadata. See “[Creating custom indexes](#)” on page 181.

## 4.17 Reindexing after removing a Documentum attribute

When you delete a custom Documentum attribute, the values for this attribute are still available in the indexes and users can find them with a full-text search. To avoid inconsistent results, reindex the object type that you modified. Rebuilding the indexes does not solve the issue since the dftxml representations for the objects still contain the attribute values. If the index agent is running in normal mode, the removal of the attribute does not trigger a new indexing. By forcing the reindexing of the objects, the index agent retrieves the objects from the repository and creates new dftxml representations.

To reindex objects of a specific type, follow the procedure described in “[Migrating documents by object type](#)” on page 92.

## 4.18 Troubleshooting the index agent

The index agent log uses DFC logger which uses log4j, and calls xPlore client API which uses slf4j. The configuration files for the two logging mechanisms are located in the WEB-INF/classes directory of the index agent WAR file. You can change the amount of information by setting the following properties:

- In logback.xml: <logger name="com.emc.documentum.core.fulltext" additivity="false" level="DEBUG">  
This package logs xPlore client information in dsearchclient.log.
- In log4j.properties: log4j.category.com.documentum.server.impl=DEBUG  
This package logs DFC information in Indexagent.log.

The log files are located in <xplore\_home>/<wildfly\_version>/server/DctmServer\_Indexagent/logs.

Indexing errors are reported in Indexagent.log. For example,

```
2012-03-28 11:16:47,673 WARN PrepWorkItem [PollStatusThread]
[DM_INDEX_AGENT_RECEIVED_FT_CALLBACK_WARN]
Received warn callback: id: 090023a380000202 message:
DOCUMENT_WARNING CPS Warning [Unknown error during text extraction(
native code: 961, native msg: access violation)].
```

### Checking indexing status

The index agent UI displays indexing status. On index agent main page, the section Detailed information of last index shows detailed information of the last index. Latest Reindex Operation Name, Total Count, Completed Count, Success Count, Warning Count and Failed Count.

When you click **Detail Details** during or after an indexing process, you see the following statistics:

- Active items: Error count, indexed content size, indexed count, last update timestamp, size, and warnings count.
- Indexer plugin: Maximum call time
- Migration progress (if applicable): Processed docs and total docs.
- Averages: Pause time, KB/sec indexed, number of indexed docs/sec, plugin blocking max time.
- List of current internal index agent threads

When you start an indexing operation, a status summary is displayed until indexing has completed. Click **Refresh** to update this summary. The summary disappears when indexing has completed.

The following table compares the processing counts reported by the index agent and xPlore administrator.

**Table 4-4: Comparing index agent and xPlore administrator indexing metrics**

Metric	Index agent	xPlore administrator
Failed	Documents not submitted to xPlore	Errors in CPS processing, content and metadata not indexed. Count does not include failures of the index agent.
Warning	Metadata indexed but not content	Metadata indexed but not content
Success	Documents indexed by xPlore	Documents indexed by xPlore

### Index agent timeout

DM\_INDEX\_AGENT\_ITEM\_TIMEOUT errors in the index agent log indicate that the indexing requests have not finished in the specified time (runaway\_item\_timeout). It does not mean that the documents were not indexed. These indexing requests are stored in the xPlore processing queue. You can also have index agent timeouts for large files. To increase text (content) and file size maximums, change the values of configuration for max\_text\_threshold, request\_time\_out, and max\_data\_per\_process.

Run the xPlore administrator report *Document Processing Error Summary* to see timeouts. For these timeouts, change the following values in `indexagent.xml`. This file is located in `<xplore_home>/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`.

- *exporter/content\_clean\_interval*: Increase the value of this parameter.
- *indexagent\_instance/runaway\_item\_timeout*: In migration mode, set this parameter to the same value as *content\_clean\_interval*.

In normal mode, also set *runaway\_item\_timeout* in the dm\_ftindex\_agent\_config object. For example, using iAPI:

```
retrieve,c,dm_ftindex_agent_config
set,c,1,runaway_item_timeout
<value>
save,c,1
```

Where *<value>* is the timeout in seconds.

- *request\_timeout\_sec*: Add this parameter to the file `indexagent.xml` and increase its value from the default 20 min. (1200 sec) to 30 min (1800 sec). The parameter is not exposed by default because of the potential out-of-memory error.

To add this parameter to `indexagent.xml`, insert the following under the `indexer_plugin_config/generic_indexer>parameter_list` element.

```
<parameter> <parameter_name>dsearch_qrserver_protocol</parameter_name>
<parameter_value>HTTP</parameter_value> </parameter>
```



**Note:** Increasing the timeout values can cause an out-of-memory error. It is recommended that you test the system under load to make sure your changes are safe.

## Submitting objects for reindexing

You can submit for reindexing the list of objects that *ftintegrity* generates ([“Using ftintegrity” on page 94](#).) To check the status of queue items that have been submitted for reindexing, navigate to the Refeed Tasks page in the index agent UI. For earlier xPlore versions, use the following DQL. For *username*, specify the user logged in to the index agent UI and start reindexing.

```
? ,c,select task_name,item_id,task_state,message from dmi_queue_item where
name=username and event='FT re-index'
```

If *task\_state* is *done*, the message is Successful batch... If the *task\_state* is *failed*, the message is Incomplete batch...

*To resubmit one document for reindexing*

Put the object ID into a temporary text file. Use the index agent UI to submit the upload: Select the **Object File** option.

*To remove queue items from reindexing*

Navigate to Refeed Tasks page in the index agent UI. For earlier xPlore versions, use the following DQL. For *username*, specify the user logged in to the index agent UI and start reindexing.

```
? ,c,delete dmi_queue_item object where name=username and
event='FT re-index'
```

## ACL and group objects are not replicated in xPlore

If the administrator has disabled full-text events for the full-text user on these objects, they are not replicated to xPlore. To verify that the full-text user is registered to generate events, run the following query:

```
? ,c,select registered_id, event from dmi_registry where user_name = 'dm_fulltext_index_user'
```

You should see a result like the following:

registered_id	event
030000f280000105	dm_move_content
030000f280000105	dm_checkin
030000f280000105	dm_readonlysave
030000f280000105	dm_destroy
030000f280000104	dm_save
030000f280000104	dm_destroy
030000f280000105	dm_save
030000f280000101	dm_save
030000f280000101	dm_destroy
030000f280000101	dm_saveasnew

## Checkout events are not registered for indexing

By default dm\_checkout events do not generate indexing. Checkout-related changes like r\_lock\_owner and r\_lock\_machine are not updated in xPlore. For example, search results do not display content as checked out. You can register dm\_checkout for indexing, but it has a performance impact.

## Documentum Server cannot find the index agent on a different domain

The index agent installer does not ask for the host name of the index agent host. Instead, it uses InetAddress.getHostName and registers the result in the <dm\_server\_config/app\_server\_uri> attribute. This is not a fully qualified domain name (FQDN). To work around this issue, manually update the value in dm\_server\_config with an FQDN value.

## Indexing is slow

The bottleneck can be in the RDBMS, index agent, or xPlore.

- If a simple SQL query like count(\*) from the sysobject table takes a long time, the RDBMS is slow. Try a restart or run update statistics.
- If the index agent log shows CONNECTOR\_PAUSED or EXPORTER\_PAUSED, the problem is in the index agent.
- If the index agent log show INDEXER\_PAUSED, the problem is in xPlore.

## Automatically stop indexing after errors

When the index agent receives a response from xPlore, a counter is updated for each error message. When the counter exceeds a configurable error\_threshold, the index agent performs the configured action, and stops indexing. To edit the error thresholds, stop the index agent instance and edit the file `indexagent.xml`. (This file is located in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`.) Locate the element `error_configs`, just after the closing tag of `indexer_plugin_config`.

For example, to automatically stop the index agent when the number of CONNECTION FAILURE errors reach 10 in 300 seconds, add the following to `indexagent.xml`:

```
<error_config>
  <error_code>CONNECTION FAILURE</error_code>
  <error_threshold>10</error_threshold>
  <time_threshold>300</time_threshold>
  <action>stop</action>
</error_config>
```

Each `error_config` element contains the following elements:

**Table 4-5: Index agent error configuration**

Element name	Description
<code>error_config</code>	Contains <code>error_code</code> , <code>error_threshold</code> , <code>time_threshold</code> and <code>action</code> elements.
<code>error_code</code>	See “ <a href="#">Error codes</a> ” on page 115.
<code>error_threshold</code>	Number of errors at which the action is executed.
<code>time_threshold</code>	Time in seconds at which to check the counter. If <code>error_threshold</code> is exceeded, the action is executed.
<code>action</code>	Valid value: stop

**Table 4-6: Error codes**

error_code	Description
UNSUPPORTED_DOCUMENT	Unsupported format
XML_ERROR	XML parsing error for document content
DATA_NOT_AVAILABLE	No information available
PASSWORD_PROTECTED	Password protected or document encrypted
MISSING_DOCUMENT	RTS routing error
INDEX_ENGINE_NOT_RUNNING	xPlore indexing service not running
CONNECTION FAILURE	xPlore server is down

By default, if xPlore server is down (CONNECTION FAILURE error), the indexing and the data ingestion stop after the specified number of errors happens in the specified time period. In this case, the status of the index agent connector and indexer threads displayed is “finished”. When the problem is solved, use the index agent UI to stop and restart the index agent.

## **DM\_SYSOBJECT\_E\_CANT\_SAVE\_NO\_LINK ERROR**

The error in the index agent log is “Cannot save xxx sysobject without any link.” Possible causes are:

- The index agent configurator failed to retrieve full-text repository objects.
- The index agent installation user does not have a default folder defined in the repository, or the folder no longer exists.

To verify, dump the user with the following iAPI commands. Substitute the installation owner name.

```
retrieve,c,dm_user where user_name='installation_owner'  
get,c,1,default_folder
```

### **4.18.1 Cleaning up the index queue**

In Documentum Administrator, you can check the index agent queue. Navigate to **Administration > Indexing Management > Index Queue**. The drop-down list displays Indexing failed, Indexing in progress, Awaiting indexing, Warning, and All. From the **Indexing failed** display, you can find the object ID and type, and the type of failure. Some types of errors are the following:

- [DM\_FULLTEXT\_E\_SEARCH\_GET\_ROW\_FAIL...] Caused by incorrect query plugin
- [DM\_FULLTEXT\_E\_QUERY\_IS\_NOT\_FTDQL...] Caused by incorrect query plugin
- [DM\_FULLTEXT\_E\_EXEC\_XQUERY\_FAIL...] There is nothing in the index.

To sort by queue state when there is a large queue, use the following DQL command in Documentum Administrator:

```
? ,c,select count(*), task_state from dmi_queue_item where name like '%fulltext%'  
group by task_state
```

To check the indexing status of a single object, get the queue item ID for the document in the details screen of the index agent UI. Use the following DQL to check the status of the queue item:

```
? ,c,select task_name,item_id,task_state,message from dmi_queue_item where name=  
username and event='FT re-index'
```

## Cleaning up the index queue

You can clean up the index queue before restarting the index agent. Using iAPI in Documentum Administrator, remove all `<dmi_queue>`\_items with the following command, inserting the full-text user for the value of name:

```
? ,c,delete dmi_queue_item object where name = 'dm_fulltext_index_user'
```

To check registered types and the full-text user name, use the following iAPI command.

```
? ,c,select distinct t.name, t.r_object_id, i.user_name from dm_type t,  
dmi_registry i where t.r_object_id = i.registered_id and i.user_name like  
'%fulltext%'
```

You see results like the following:

name	r_object_id	user_name
dm_group	0305401580000104	dm_fulltext_index_user
dm_acl	0305401580000101	dm_fulltext_index_user
dm_sysobject	0305401580000105	dm_fulltext_index_user

## 4.18.2 Index agent startup issues

### Firewall issues

The following error is logged in indexagent.log:

```
com.xhive.error.XhiveException [TRANSACTION_STILL_ACTIVE]  
The operation can only be done if the transaction is closed
```

Enable connections between the index agent host, the Documentum Server, and xPlore through the firewall.

### Startup problems

Make sure that the index agent web application is running. On Windows, verify that the Documentum Indexagent service is running. On Linux, verify that you have instantiated the index agent using the start script in `xplore_home/<wildfly_version>/server`.

Make sure that the user who starts the index agent has permission in the repository to read all content that is indexed. If the repository name is reported as null, restart the repository and the connection broker and try again. If you see a status 500 on the index agent UI, examine the stack trace for the index agent instance. If a custom routing class cannot be resolved, this error appears in the browser:

```
org.apache.jasper.JasperException: An exception occurred processing JSP page  
/action_dss.jsp at line 39  
...  
root cause  
com.emc.documentum.core.fulltext.common.IndexServerRuntimeException:  
com.emc.documentum.core.fulltext.client.index.FtFeederException:  
Error while instantiating collection routing custom class...
```

If the index agent web application starts with port conflicts, stop the index agent with the script. If you run a stop script, run as the same administrator user who

started the instance. The index agent locks several ports, and they are not released by closing the command window.

### **Restarting the index agent**

If you stop and restart the index agent before it has finished indexing a batch of documents that you manually submitted through the index agent UI, resubmit the indexing requests that were not finished.

### **Cannot stop the index agent**

If you have configured two index agents on the same host and port, you see the following error message when you attempt to stop the agent:

```
Exception in thread "main" java.lang.SecurityException:  
Failed to authenticate principal=admin, securityDomain=jmx-console
```

You can kill the JVM process and run the index agent configurator to give the agents different ports.

#### **4.18.3 Content query returned no data**

If you see the warning “Content query returned no data” in the index agent log, the index agent was not able to get the content for the file with the specified ID. The object is not indexed and remains in the indexing queue. For example:

```
Content query returned no data for object  
09026d9180049404 DfFtExportContext.java:363
```

Possible causes: I/O error, file permissions, or network packet failure.

# Chapter 5

## Document Processing (CPS)

### 5.1 About CPS

The content processing service (CPS) performs the following functions:

- Retrieves indexable content from content sources
- Determines the document format and primary language
- Parses the content into index tokens that xPlore can process into full-text indexes

If you test Documentum indexing before performing migration, first replicate security. See “[Manually updating security](#)” on page 64. For information on customizations to the CPS pipeline, see “[Custom content processing](#)” on page 155.

#### Language identification

Some languages have been tested in xPlore. Many other languages can be indexed. Some languages are identified fully including parts of speech, and others require an exact match. For a list of languages that CPS detects, see Basistech documentation. If a language is not listed as one of the tested languages in the xPlore release notes, search must be for an exact match. For tested languages, linguistic features and variations that are specific to these languages are identified, improving the quality of search experience.

#### White space

White space such as a space separator or line feed identifies word separation. Then, special characters are substituted with white space. See “[Handling special characters](#)” on page 138.

For Asian languages, white space is not used. Entity recognition and logical fragments guide the tokenization of content.

#### Case sensitivity

All characters are stored as lowercase in the index. For example, the phrase “I'm runNiNg iN THE Rain” is lemmatized and tokenized as “I be run in the rain.” There is a limited effect of case on lemmatization. In some languages, a word can have different meanings and thus different lemmas depending on the case. Case sensitivity is not configurable.

## 5.2 Adding a remote CPS instance

By default, every xPlore instance has a local CPS service. Each CPS service receives processing requests on a round-robin basis. For a high-volume environment with multiple xPlore instances, you can configure dedicated CPS instances for instances identified as bottlenecks. You must have low network latency. This dedicated CPS reduces network overhead. See “[Configuring CPS dedicated to indexing or search](#)” on page 122.

To improve indexing or search performance, you can install CPS on a separate host. The installer adds a WildFly instance, CPS *war* file, and CPS native daemon on the remote host.



**Note:** The remote instance must be on the same operating system as other xPlore instances.

1. Install the remote CPS instance using the xPlore installer.
2. Configure the instance for CPS only.
3. Start the instance using the start script `startRemoteCPSInstanceName.cmd` or `startRemoteCPSInstanceName.sh` in `xplore_home/wildfly_version/server`. (On Windows, the standalone instance is installed as an automatic service.)
4. Register the remote CPS instance in xPlore administrator. Open **Services > Content Processing Service** in the tree.
5. In the **Content Processing Service** page, click Add.
6. In the Add Service window, select Remote and provide information of the remote CPS instance you are adding.

- a. Enter the URL to the remote instance using the following syntax:

```
http://hostname:port/cps/ContentProcessingService?wsdl
```

- b. From the Instance list, select an instance you want to add the CPS to.
- c. From the Usage list, specify whether the CPS instance processes indexing requests (the *index* option), search requests (the *search* option), or both (the *all* option).
- d. Click OK.



**Note:** The added CPS takes effect immediately.

7. Specify whether the CPS instance performs linguistic processing (lp) or text extraction (te). If a value is not specified, TE and LP are sent to CPS as a single request.
  - a. In `indexserverconfig.xml`, locate the `content-processing-services` element. This element identifies each CPS instance. The element is added when you install and configure a new CPS instance.

- b. Add or change the capacity attribute on this element. The capacity attribute determines whether the CPS instance performs text extraction, linguistic processing, or all. In the following example, a local CPS instance analyzes linguistics, and the remote CPS instance processes text extraction. For example:

```
<content-processing-services context-characters="!.,;?'&quot;" special-characters="@#$%^~`*&gt;:()-+=&lt;"/[]{}">
<content-processing-service capacity="lp" usage="all" url="local"/>
<content-processing-service capacity="te" usage="index" url="http://myhost:9700/cps/ContentProcessingService?wsdl"/>
</content-processing-services>
```

8. Wait to ensure the change has been applied to all xPlore instances.
9. Test the remote CPS service using the WSDL testing page, with the following syntax:

`http://hostname:port/cps/ContentProcessingService?wsdl`

After you install and register the remote instance, you see it in the Content Processing Service UI of xPlore administrator. You can check the status and see version information and statistics.

Check the CPS daemon log file `cps_daemon.log` for processing event messages. For a local process, the log is in `xplore_home/<wildfly_version>/server/DctmServer_NodeInstanceName/logs`. For a remote CPS instance, `cps_daemon.log` is located in `cps_home/<wildfly_version>/server/DctmServer_RemoteCPSInstanceName/logs`. If a CPS instance is configured to process text only, TE is logged in the message. For linguistic processing, LP is logged.

## 5.3 Removing a CPS instance

By default, every xPlore instance has a local CPS service. After you add a remote CPS to the local xPlore instance, you can disable or remove the local CPS so that the local xPlore instance can leverage extra CPU and memory capacity on the remote host.

To remove an existing CPS instance, in the Content Processing Service page, click the Remove action button (red cross) next to the CPS you want to remove. Confirm that the CPS has been removed.



**Note:** An xPlore instance must have at least one CPS configured for it. If an xPlore instance has only one CPS, either local or remote, you cannot disable or remove it.

## 5.4 Configuring CPS dedicated to indexing or search

By default, every xPlore instance has a local CPS service. All CPS services receive processing requests on a round-robin basis. For high-volume environments with multiple xPlore instances, you can configure one or more CPS services to handle all processing requests for a specific xPlore instance.

You can configure additional CPS instances that are dedicated to indexing in xPlore administrator, for high ingestion requirements, or dedicated to search, for heavy search usage.

1. Expand **Services** in the left panel, click **Content Processing Service**.
2. Click **Add** on the Content Processing Service page.
3. Select **local or remote instance** in the **Add Service** window, and then select **node instance and usage** in the drop-down list. For remote instance, you must specify its URL. Remote CPS service is always configured for dedicated indexing or search for specific xPlore instances.

Make sure that you have a remaining global CPS instance. For example:

```
<content-processing-services context-characters="!,.;?'&quot;"  
    special-characters="@#$%^~`*&gt;:()-.+=&lt;&gt;/[]{}">  
    <content-processing-service usage="all" url="local"/>  
</content-processing-services>
```

## 5.5 Administering CPS

### Configuring Starting and stopping CPS

You can configure CPS properties, stop and start CPS in xPlore administrator.

To operate on remote CPS, expand **Services**, expand **Content Processing Service**. Remote CPS services are displayed under **Content Processing Service**, for example, <http://host:port/cps/ContentProcessingService?wsdl>.

To operate on local CPS, expand **Instances**, expand an xPlore node, and then expand **Content Processing Service**. The local service is displayed under **Content Processing Service**.

1. Configure CPS properties: Select a remote or local service, click **Configuration** on the up-right corner. The **CPS Configuration** window appears. Modify properties in this window.
2. Stop CPS: Select a remote or local service, click **Stop CPS** on the up-right corner, and then click **Suspend**.
3. Start CPS: Select a remote or local service, click **Start CPS** and then click **Resume**. Select an instance in the xPlore administrator tree, expand it, and choose **Content Processing Service**. Click **Start CPS** and then click **Resume**.

If CPS crashes or malfunctions, the CPS manager tries to restart it to continue processing. If the restart fails, check the corresponding CPS configuration file to ensure the configuration is appropriate.

### CPS status and statistics

To view all CPS instances in the xPlore federation, expand **Services > Content Processing Service**. For more information on remote instances, see [“Adding a remote CPS instance” on page 120](#).

To view version information and statistics about a specific CPS instance, click a remote or local service under **Content Processing Service**.

## 5.6 Modifying CPS configuration file

You can modify some linguistic or text extraction parameters in CPS configuration file.

The following procedure provides high-level steps that you must perform for any change in CPS configuration file.

1. Stop all CPS instances.
2. Make a copy of the CPS configuration file `NodeInstanceName_local_configuration.xml` located in `<xplore_home>/dsearch/cps/cps_daemon`.
3. Edit the CPS configuration file.
4. Make the required changes using an XML editor. Changes must be encoded in UTF-8. Do not use a simple text editor such as Notepad, which can insert characters using the native OS encoding and cause validation to fail.
5. Make the same changes in the configuration files of all remote CPS instances. The configuration file of a remote CPS instance is named `RemoteCPSInstanceName_configuration.xml` and located at `<xplore_home>/dsearch/cps/cps_daemon`.

Remember to make a copy of the configuration files before modifying them.

6. Start all CPS instances.

## 5.7 Maximum document and text size

This section describes the configuration properties related to the size of documents that are indexed. Depending on your documents, modifying these settings can improve or impact the ingestion performance. To index large documents concurrently, implement one of the following solutions:

- Add more memory and CPU capacity and stop unnecessary processes that compete for resources.
- Add CPS daemons as described in “[Adding CPS daemons for ingestion or query processing](#)” on page 146. Only add CPS daemons if xPlore host has enough memory and CPU capacity.
- Add remote CPS instances as described in “[Adding a remote CPS instance](#)” on page 120.

If xPlore host has enough memory and CPU capacity, you can add a CPS instance to the same host, as described in *Adding CPS instances in OpenText Documentum xPlore Installation Guide*.

- Enable retry in a separate daemon for documents that failed during CPS processing.
  1. Edit the CPS configuration file `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` located in `<xplore_home>/dsearch/cps/cps_daemon`.
  2. Edit the `retry_failure_in_separate_daemon` property and set it to `true`. Default: `True`.

This can have a small impact on performance.

- To avoid query timeouts when the indexing load is high, enable search on a dedicated CPS daemon:
  1. Edit the CPS configuration file `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` located in `<xplore_home>/dsearch/cps/cps_daemon`.
  2. Edit the `query_dedicated_daemon_count` property and set the number of daemons dedicated to searches. Default: 1.

### Maximum document size

Indexing agent (Documentum only) limits the size of the documents submitted for indexing. Stop the index agent instance to change the size limit. Set the maximum document size in the index agent configuration file `indexagent.xml`, which is located in:

```
<indexagent_home>/<wildfly_version>/server/DctmServer_IndexAgentInstanceName/deployments/IndexAgent.war/WEB-INF/classes
```

Edit the *contentSizeLimit* parameter within the parent element *exporter*. The value is in bytes. Default: 20000000. Larger documents are skipped. Only their metadata is indexed.

## Maximum text size per document

CPS limits the size of text within a document that is indexed. A document can have a much greater size (*contentSizeLimit*) compared to the indexable text within the document.

Set the maximum size of text within a document and the text in CPS batch in CPS configuration. Choose an instance in xPlore administrator and click **Configuration**. Edit the *max text threshold* parameter to set the size limit, in bytes. Default: 10485760 (10 MB). Maximum setting: 2 GB. Larger values can slow ingestion rate and cause more instability. Includes expanded attachments. For example, if an email has a zip attachment, the zip file is expanded to evaluate document size. If you increase this threshold, ingestion performance can degrade under heavy load. If a document content text size is larger than this value and if the *cut off text* parameter is set to false, CPS indexes only the metadata, not the content. Other documents in the same batch are not affected. Increasing the maximum text size can negatively affect CPS memory consumption under heavy load. In this case, the entire batch of submitted documents fails.

## Partially index large documents (cut off text)

You can configure CPS to cut off text in documents that exceed *max text threshold*. When configured, CPS indexes part of the content, up to the threshold instead of only the metadata. Set *cut off text* to true in  
NodeInstanceName\_local\_configuration.xml or  
RemoteCPSInstanceName\_configuration.xml (default: false). This file is located in  
<xplore\_home>/dsearch/cps/cps\_daemon.

Documents that are partially indexed are recorded in cps\_daemon.log: *docxxxx is partially processed*. The dftxml is annotated with the attribute *partialIndexed* on the dmftcontentref element.

## Maximum text size in CPS batch

CPS processes documents by batches. You can configure the upper limit for a batch of documents in CPS processing. Incoming requests are put on hold if the total content size for all documents in a CPS batch exceeds the limit.

Edit the CPS configuration file NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml located in <xplore\_home>/dsearch/cps/cps\_daemon. Edit *max\_data\_per\_process* and set its value in bytes. Default: 30 MB. Maximum setting: 2 GB.

For a hypothetical example, with the default of 30 MB, 3 documents with 10-MB text, 30 documents of 1-MB text, or 300 documents of 100 KB of text would fill up the batch. If the batch text size is exceeded, all documents in the batch fail. You can refeed them through the index agent.

## Embedded content used for index rebuild

Set the maximum in xPlore administrator. Choose **Indexing Service** in the tree and click **Configuration**. Edit the value of *rebuild-index-embed-content-limit*. Below this value, content is embedded in requests sent to CPS. It is used for rebuilding the index. Larger content is passed in a file, not embedded. The file is added to the *rebuild-index-temp-path*, which is configured in *indexserverconfig.xml > index-config*. If that property does not exist, it is added to the export folder of the first CPS..Default: 2048 bytes.

## Embedded XML content maximum size when index-as-sub-path is true

Set the maximum size of XML content in byte that is embedded into dftxml. Edit *indexserverconfig.xml* and set the value of the *file-limit* attribute in the *xml-content* element. Default: 512KB.

If zone search is enabled and there are a lot of XML files in the repository, query with summary can be a little slower than other file formats as summary processing has to concatenate all element texts before computing.

## Very large documents

In order to process a very large file (more than 100 MB), you can temporarily enlarge several thresholds in *indexagent.xml* and CPS  
*NodeInstanceName\_local\_configuration.xml* or  
*RemoteCPSInstanceName\_configuration.xml*. In CPS configuration:  
*max\_text\_threshold*, *request\_time\_out*, *max\_data\_per\_process*. In Index Agent configuration: *runaway\_item\_timeout*, *content\_clean\_interval*, *request\_timeout\_sec*, and *parameter\_list* (this property needs to be manually added in *indexer\_plugin\_config ->generic\_indexer*). These changes will degrade xPlore performance.

## 5.8 Configuring languages and encoding

Some languages have been tested in xPlore. (See the release notes for this version.) Many other languages can be indexed. Some languages are identified fully including parts of speech, and others are restricted to an exact match. For the list of identified languages and encodings for each language, see the Basistech documentation.

### Configuring language identification

The language of the content determines how the document is tokenized. During indexing, CPS identifies the language of the document. If your repository has a single language, you can override this behavior by setting a default locale for indexing. You can set a default locale for indexed content and one for metadata. You can also set a default locale for queries.

To force a default indexing locale, add the following property to every category definition or domain definition in *indexserverconfig.xml*. Settings in the domain

definition takes precedence over those in the category definition. These settings turn off language identification. Do it only if your repository has a majority of documents in a single language. Use ISO-639 language code for the value attribute:

Set a default indexing locale in a category:

```
<category name="dftxml">
<properties>
  ...<property value="en" name="index-default-locale"/>
</properties>
.....
</category>
```

Set a default indexing locale in a domain:

```
<domain>
<properties>
  ...<property value="fr" name="index-default-locale"/>
</properties>
.....
</domain>
```

To force a different default locale for metadata, add a property to every category definition or domain definition like the following. Settings in the domain definition takes precedence over those in the category definition.

Set a default locale for metadata in a category:

```
<category name="dftxml">
<properties>
  ...<property value="fr" name="index-metadata-default-locale"/>
</properties>
.....
</category>
```

Set a default locale for metadata in a domain:

```
<domain>
<properties>
  ...<property value="en" name="index-metadata-default-locale"/>
</properties>
.....
</domain>
```

For a query, the session locale from one of the following is used as the language for linguistic analysis:

- Webtop login locale
- DFC API locale or dfc.properties
- iAPI and iDQL Documentum Server locale

The language indexing object is identified by content or metadata:

- Content

CPS uses the first 65536 characters in the document to identify the language of the document.

- Metadata

By default, CPS uses the following Documentum object attributes to identify the language of the metadata: object\_name, title, subject, and keywords, which is configured in indexserverconfig.xml > category > element-for-language-identification.

To set query locale:

1. Configure a default language if one is not identified. The default language must be one of the supported languages listed in the linguistic\_processor element of the file NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml. This file is located in *xplore\_home/dsearch/cps/cps\_daemon*.
2. Open indexserverconfig.xml in *xplore\_home/config*.
3. Modify the property named *query-default-locale* with the desired language under the *search-config* element. For example:

```
<search-config><properties>
  <property value="en" name "query-default-locale"/>...
```

The query locale can be overridden by setting the property *dsearch\_override\_locale* in *dm\_ftengine\_config*.

4. Change the metadata that are used for language identification. Set an attribute as the value of the name on the element element-for-language-identification. For example:
5. Validate your changes to indexserverconfig.xml. See “[Modifying indexserverconfig.xml](#)” on page 54.

```
<linguistic-process>
  <element-for-language-identification name="object_name" />
  <element-for-language-identification name="title" />
  <element-for-language-identification name="subject" />
  <element-for-language-identification name="keywords" />
</linguistic-process>
```

6. (Optional) Check the identified language for a document: Use xPlore administrator to view the dftxml of a document. Click the document in the collection view, under **Data Management**. The language is specified in the *lang* attribute on the *dmftcontentref* element. For example:

```
<dmftcontentref content-type="" lang="en" encoding="utf-16le" ...>
```

7. (Optional) Check the session locale for a query. Look at the xPlore log event that prints the query string (in *dsearch.log* of the primary xPlore instance). The event includes the query-locale setting used for the query. For example:

```
query-locale=en
```

8. (Optional) Change the session locale of a query. The session\_locale attribute on a Documentum object is automatically set based on the OS environment. To search for documents in a different language, change the local per session in DFC or iAPI. The iAPI command to change the session\_locale:

```
set,c,sessionconfig,session_locale
```

The DFC command to set session locale on the session config object (IDfSession.getSessionConfig):

```
IDfTypedObject.setString("session_locale", locale)
```

## Overriding language identification for metadata

In some situations, language used by metadata and content should be consistent but the language identified for metadata does not always represent the language used by the corresponding content. To avoid this, suppress language identification for metadata. CPS always use the same language identified for content.

In indexserverconfig.xml, set the following property value to True under the index-config element. Add the property if not present in the configuration file.

```
<property value="true" name="use-content-lang-for-metadata"/>
```

For multi-language documents, set this option to false.

The index-metadata-default-locale property overrides the use-content-lang-for-metadata property in indexserverconfig.xml. For example, if you set index-default-locale to "en", index-metadata-default-locale to "fr", and use-content-lang-for-metadata to "true", CPS uses French as the language for metadata rather than English.



**Note:** If the extraction configuration option xml-content index-as-sub-path is set to true, the use-content-lang-for-metadata does not take effect.

Add or update the following properties to the search-config element in indexserverconfig.xml to support language detection for queries. Query language is detected or bypassed depending on these properties

- query-enable-language-detection: Set to true to detect query language. If false, the query application server language is used.
- query-no-detection-input-language: If the session locale matches one of the listed languages then the detection is not attempted.

For example, the user session locale is in English, but the user queries in multiple languages. English (en) should not be listed in query-no-detection-input-language. It allows auto language detection. If a user always queries in the same language as the session locale, then add the language to the list. It prevents the document language from being identified incorrectly.

- query-detection-language-override: List the languages that override the input language when they are detected. If the detected language matches one of the languages listed, the query application server language is overridden with the detected language.
- query-expansion-lang-list: List the languages that enable queries with multiple languages. Use ISO-639 to represent languages and split with a comma.
- query-language-detection-boost-language-list: Specifies a list of languages to boost in the query phase during query language detection in the following pattern:

```
<property value="language_code_1[weight_1],language_code_2[weight_2], ...
language_code_n[weight_n]"
name="query-language-detection-boost-language-list"/>
```

The weight in [] is an integer indicating the possibility of being chosen as the query language. The higher the weight, the more likely the corresponding language will be chosen. For example, in the following setting, both Chinese and Japanese are boosted during query language detection and Chinese is more likely to be selected as the query language.

```
<property value="zh[10],ja[2]" name="query-language-detection-boost-language-list"/>
```

After you change these properties, restart all xPlore instances.

## Handling apostrophes

Some languages have more apostrophes as part of a name or other part of speech. The default list of special context characters includes the apostrophe. Apostrophes in words are treated as white spaces. You can remove the apostrophe from the list if words are not correctly found when you run a search. See “[Handling special characters](#)” on page 138.

## Supporting bidirectional languages

Bidirectional languages such as Arabic or Hebrew are mostly written from right to left, while some portions are written from left to right. You must configure the following property to support bidirectional languages for files in PDF format. Do not set this property if your environment does not index bidirectional documents, because this change impacts performance.

Edit `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` in `xplore_home/dsearch/cps/cps_daemon`. Add or edit the `reorder_bidi_pdf` property to the stellent `text_extractor` element:

```
<text_extractor>
<name>stellent</name>
...
<properties>
<property name="extract_xmp_metadata">true</property>
<property name="identify_file_normally">true</property>
<property name="read_buffer_size">2</property>
<property name="allow_sub_doc_failure">true</property>
<property name="reorder_bidi_pdf">true</property>
```

If you have already indexed objects in PDF format that would be affected by this change, you must reindex them.

## Wildcard Search in Hebrew

In Hebrew, some compound word cases can be hit without wildcard searching. For example, in Hebrew, the word “the” is a part of the word it wants to describe and represent, unlike in the English language where the word ‘the’ is considered as a separate word.

You can configure the following property to support compound word searching in Hebrew. Edit `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` in `xplore_home/dsearch/cps/cps_daemon`. Add or edit the `guess_hebrew_prefixes` property to the `Rosette_linguistic_processor` element:

```
...
<linguistic_processor>
    <name>RSE</name>
    <type>java</type>
    <lib_path>com.emc.cma.cps.processor.linguisticprocessor.CPSRSELinguisticProcessoremc.cma.
    cps.processor.linguisticprocessor.CPSRSELinguisticProcessor</lib_path>
        <!--the some properties of this process-->
        <properties>
            <property name="max_data_per_process">31457280</property>
            <property name="BT_ROOT">..//cps_manager/thirdparty/rse</property>
            <property name="guess_hebrew_prefixes">true</property>
            <property name="guess_hebrew_prefixes">true</property>
        </properties>
        <languages>he</languages>
        <!--only Hebrew-->
    </linguistic_processor>
...

```

After you update the configuration, you must reindex the existing documents.

## 5.9 Indexable formats

Some formats are fully indexed. For some formats, only the metadata is indexed. For a complete list of supported formats, refer to the documentation for *Oracle Outside In* available on the Oracle website.

If a format cannot be identified, navigate to **Reports**, select the **View as Document Processing Error Detail**, select the **Processing Error Code** as *772 – Unsupported file format* and run the report.

## 5.10 Lemmatization

### 5.10.1 About lemmatization

Lemmatization is a normalization process that reduces a word to its canonical form. For example, a word like *books* is normalized into *book* by removing the plural marker. *Am*, *are*, and *is* are normalized to “be.” This behavior contrasts with stemming, a different normalization process in which stemmed words are reduced to a string that sometimes is not a valid word. For example, *ponies* becomes *poni*. xPlore uses an indexing analyzer that performs lemmatization. Studies have found that some form of stemming or lemmatization is almost always helpful in search.

Lemmatization is applied to indexed documents and to queries. Lemmatization analyzes a word for its context (part of speech), and the canonical form of a word (lemma) is indexed. The extracted lemmas are actual words.

## Alternate lemmas

Alternative forms of a lemma are also saved. For example, *swim* is identified as a verb. The noun lemma *swimming* is also saved. A document that contains *swimming* is found on a search for *swim*.

If you turn off alternate lemmas, you see variable results depending on the context of a word. For example, *saw* is lemmatized to *see* or *to saw* depending on the context. See “[Configuring indexing lemmatization](#)” on page 133.

## Query lemmatization

Lemmatization of queries is more prone to error because less context is available in comparison to indexing.

The following queries are lemmatized:

- IDfxQuery: The *with stemming* option is included.
- The query from the client application contains a wildcard.
- The query is built with the DFC search service.
- The DQL query has a search document contains (SDC) clause (except phrases). For example, the query *select r\_object\_id from dm\_document search document contains 'companies winning'* produces the following tokens: *companies*, *company*, *winning*, and *win*.

## Phrase search and lemmatization

Even though phrases are not lemmatized, they may return a document that has been lemmatized during indexing. For example, a document contains the word *felt*. It is lemmatized to *felt* and *feel*. A phrase search for *feels happy* does not find the document. However, a search for *feel happy* finds it, even though the original phrase is *felt happy*.

You can configure xPlore to match phrases exactly in queries. This requires more space for the index, because extra exact form with special characters are stored. You must reindex documents to get this exact match for phrases. Also, if you enable this feature, fuzzy search will not work. For information about fuzzy search, see “[Configuring fuzzy search](#)” on page 264.

To configure exact phrase match in queries:

1. Edit `indexserverconfig.xml` in `xplore_home/config`.
2. Add a `query-exact-phrase-match` property to the `search-config` element:

```
<search-config>
  <properties>
    <property value="en" name="query-default-locale" />
    ...
    <property value="true" name="query-exact-phrase-match" />
  </properties>
</search-config>
```

3. Restart the primary and secondary xPlore instances.
4. Rebuild the index using xPlore administrator.

### Lemmatization and index size

Lemmatization saves both the indexed term and its canonical form in the index, effectively doubling the size of the index. Multiple alternates for the lemma also increase the size of the index. Evaluate your environment for index storage needs and lemmatization expectations for your users.

## 5.10.2 Configuring indexing lemmatization

For information on configuring lemmatization, see “[Configuring query lemmatization](#)” on page 257.

1. To turn off lemmatization for indexing, add an *enable-lemmatization* attribute to the *index-server-configuration* element in *indexserverconfig.xml*. Set the value to *false*. See “[Modifying indexserverconfig.xml](#)” on page 54.



**Note:** If you wish to apply your lemmatization changes to the existing index, reindex your documents.

2. Alternate lemmas are generated by default. To turn off alternate lemmas, modify the file *cps\_context.xml* located in *xplore\_home/dsearch/cps/cps\_daemon*. Set the value attribute on the following property to *false* (default = *true*):

```
<property name="com.basistech.bl.alternatives" value="false"/>
```

## 5.10.3 Lemmatizing specific types or attributes

By default, all input is lemmatized unless you configure lemmatization for specific types or attributes.

1. Open *indexserverconfig.xml* in *xplore\_home/config*.
2. Locate the *category* element for your documents (not ACLS and groups). Add or edit a *linguistic-process* element. This element can specify elements or their attributes that are lemmatized when indexed, as shown in the following table of child elements.

**Table 5-1: linguistic-process element**

Element	Description
element-with-name	The <i>name</i> attribute on this element specifies the name of an element that contains lemmatizable content.

Element	Description
save-tokens-for-summary-processing	Child of <i>element-with-name</i> . If this element exists, the parent element tokens are saved. They are used in determining a summary or highlighting. Specify the maximum size of documents in bytes as the value of the attribute <i>extract-text-size-less-than</i> . Tokens will not be saved for larger content. Set the maximum size of tokens for the element as the value of the attribute <i>token-size</i> .
element-with-attribute	The <i>name</i> attribute on this element specifies the name of an attribute on an element. The <i>value</i> attribute contains a value of the attribute. When the value is matched, the element content is lemmatized.
element-for-language-identification	Specifies an input element that is used by CPS to identify the language metadata of the document.

In the following example, the content of an element with the attribute *dmfttype* with a value of *dmstring* is lemmatized. These elements are in a dftxml file that the index agent generates. For the dftxml extensible DTD, see “[Extensible Documentum DTD](#)” on page 413.

If the extracted text does not exceed 262144 bytes (*extract-text-size*), the specified element is processed. In the example, an element with the name *dmftcustom* is processed. Several elements are specified for language identification.

```
<linguistic-process>
  <element-with-attribute name="dmfttype" value="dmstring"/>
  <element-with-name name="dmftcustom">
    <save-tokens-for-summary-processing extract-text-size-less-than="262144" token-size="65536"/>
  </element-with-name>
  <element-for-language-identification name="object_name"/> ...
</linguistic-process>
```

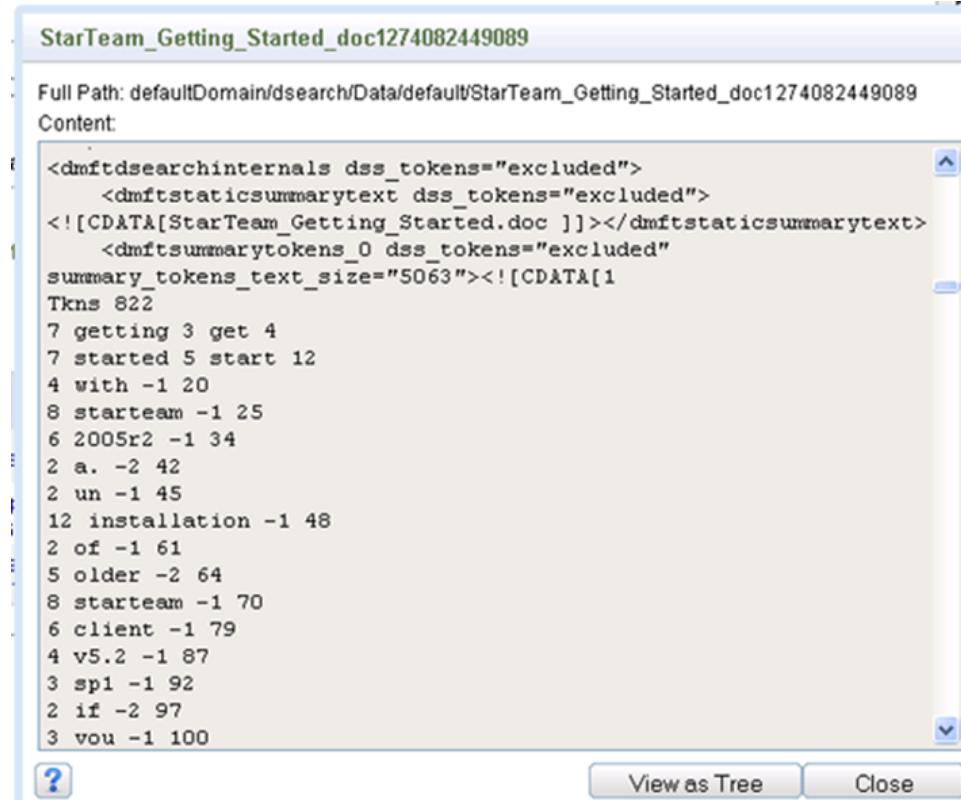


**Note:** If you wish to apply your lemmatization changes to the existing index, reindex your documents from the IndexAgent.

#### 5.10.4 Troubleshooting lemmatization

If a query does not return expected results, examine the following:

- Test the query phrase or terms for lemmatization and compare to the lemmatization in the context of the document. (You can test each sample using xPlore administrator *Test Tokenization*.)
- View the query tokens by setting the dsearch logger level to DEBUG using xPlore administrator. Expand **Services > Logging** and click **Configuration**. Set the log level for *dsearch-search*. Tokens are saved in *dsearch.log*.
- Check whether some parts of the input were not tokenized because they were excluded from lemmatization: Text size exceeds the configured value of the *extract-text-size-less-than* attribute.
- Check whether a subpath excludes the dftxml element from search. (The subpath attribute *full-text-search* is set to false.)
- If you have configured a collection to save tokens, you can view them in the xDB admin tool. (See “[Debugging queries](#)” on page 318.) Token files are generated under the Tokens library, located at the same level as the Data library. If *save-tokens-for-summary-processing* is enabled for one element, you can also view tokens in the stored dftxml using xPlore administrator. The number of tokens stored in the dftxml depends on the configured amount of tokens to save. To see the dftxml, click a document in a collection.



The screenshot shows a window titled "StarTeam\_Getting\_Started\_doc1274082449089". The content pane displays XML code representing tokens. The code includes XML tags like <dmftdsearchinternals>, <dmftstaticsummarytext>, and <dmftsummarytokens\_0>. Below the XML, a list of tokens is shown with their counts, such as "Tkns 822", "7 getting 3", "7 started 5", etc. At the bottom of the window are buttons for "?", "View as Tree", and "Close".

```

<dmftdsearchinternals dss_tokens="excluded">
    <dmftstaticsummarytext dss_tokens="excluded">
        <![CDATA[StarTeam_Getting_Started.doc ]]></dmftstaticsummarytext>
        <dmftsummarytokens_0 dss_tokens="excluded">
            summary_tokens_text_size="5063"><![CDATA[1
Tkns 822
7 getting 3 get 4
7 started 5 start 12
4 with -1 20
8 starteam -1 25
6 2005r2 -1 34
2 a. -2 42
2 un -1 45
12 installation -1 48
2 of -1 61
5 older -2 64
8 starteam -1 70
6 client -1 79
4 v5.2 -1 87
3 sp1 -1 92
2 if -2 97
3 you -1 100

```

**Figure 5-1: Tokens in dftxml**

### 5.10.5 Saving lemmatization tokens

You can save the tokens of metadata and content. Tokens are used to rebuild the index.

1. Open indexserverconfig.xml in *xplore\_home/config*.
2. Set the property *save-tokens* to true for the collection. The default is *false*.

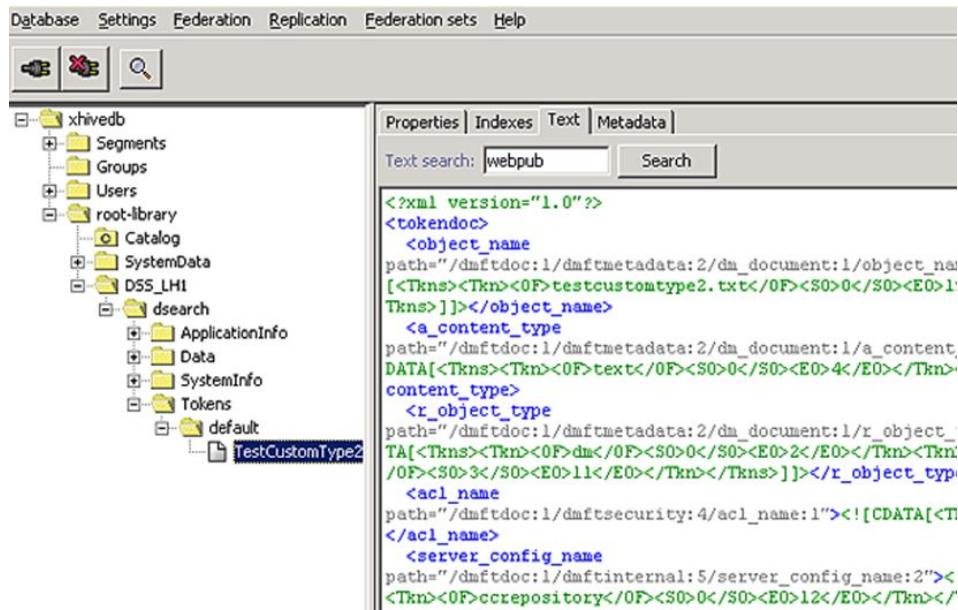
For example:

```

<collection document-category="dftxml" usage="Data" name="default">
    <properties>
        <property value="true" name="save-tokens" />
    </properties>
</collection>

```

3. You can view the saved tokens in the xDB tokens database. Open the xDB admin tool in *xplore\_home/dsearch/xhive/admin*.



**Figure 5-2: Tokens in xDB**

The tokens database stores the following tokens:

- Original and root forms (lemmas) of the text
- Alternate lemmas
- The components of compound words
- The starting and ending offset relative to the field the text is contained in
- Whether the word was identified as a stop word

### 5.10.6 Defining disambiguation rules

To ensure correct lemmatization of ambiguous terms, you can define one or more disambiguation rules and add rules files to the following directory:

```
 ${xplore_home}/dsearch/cps/cps_manager/thirdparty/rse/disambig_rules/
 custom
```

 **Note:** Disambiguation rules are currently designed for Hebrew. Add "he\_" as prefix to rules files to post-process Hebrew tokens.

#### Rules format

Disambiguation rules are based on the concepts of original form and root form of terms. The original form of a term can have variants, whereas the root form is the basic linguistic unit, or lemma. For example, *run*, *runs*, *ran*, and *running* are variants of the root form *run*.

A disambiguation rule starts with a + or - rule sign, followed by one original form, and then by one or more root forms. If the rule sign is +, root forms (lemmas) are added to the original form. If it is -, root forms are removed from the original form.

The rule is applied to the original form, or first word after the rule sign. The subsequent words are the root forms that you want to add or remove. In the rules file, a line starting with "#" is considered as a common line.

### Rules compatibility and priority

To apply more than one disambiguation rule, the rules need to be compatible. When a rule, or part of it, has the same original form as another rule, they require opposite rule signs.

There are two types of rules files: default and custom. Default rules files are located in \${xPlore\_home}/dsearch/cps/cps\_manager/thirdparty/rse/disambig\_rules/default/. Default rules are upgraded when patches are applied and should not be edited.

The following rule priorities resolve conflicts:

- When custom rules are in conflict with default rules, custom rules override default rules.
- If two rules are in conflict, xPlore will execute minus ("−") rules rather than plus ("+") rules.

### Defining advanced rules

In some cases, you may need to define more complex disambiguation rules. For example, assume "ABC" represents a term with lemmas "A", "B" and "C". A query for "ABC" might return a term such as "BX", which has lemmas "B" and "X". In this case, a rule "-ABC BX" will not take effect, since "ABC" has no lemma "BX". You need to define a rule "-ABC B" or "-BX B".

You can determine requirements for such rules by testing tokenization. Enter words in root form and original form and, if a wrong lemma is returned, define an appropriate rule. See "[Testing tokenization](#)" on page 144.

## 5.11 Handling special characters

Special characters include accents in some languages. Accents are removed to allow searches for words without supplying the accent in the term. You can disable diacritics removal. Other special characters are used to break text into meaningful token: characters that are treated as white space, and punctuation.

If the following Unicode characters are contained within a word, they can be indexed. Words that contain these characters 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 characters 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

## Handling integer tokenization

By default, integers with spaces or dashes generate individual tokens in some languages, like English, but not in others, like German. For example, 500–599 generates two tokens in English (500 and 599) but only one in German (500 599). A search for 500 from a German locale does not find the document with 500–599 in the object name.

To generate individual tokens, do the following: from xPlore administrator > Services > Content Processing service > CPS plugin configuration, edit the CPS configuration file `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` located in `xplore_home/dsearch/cps/cps_daemon`, locate element `linguistic_processing/properties/property`, and add the following:

```
<property name="force_tokenize_on_whitespace">true</property>
```

Note that turning on this property can have an impact on ingestion and search performance.

## Handling words with accents (diacritics)

Words with accents, such as those in English, French, German, and Italian, are indexed as normalized tokens (accents removed) to allow search for the same word without the accent. However diacritics are not removed from a query regardless of the `normalize_form` setting when the term contains a wildcard.

You can prevent diacritics from being normalized, in this case, only exact search terms (with accents) return results. For example, the search term `chateau` would not return objects with `château`.

You can also index the two forms: the original form and the normalized form to allow wildcard searches. In this case, `normalize_form` must be set to true.

*To prevent diacritics from being normalized, do the following from xPlore administrator > Services > Content Processing service > CPS plugin configuration:*

1. Change diacritics removal in the CPS configuration file `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` located in `xplore_home/dsearch/cps/cps_daemon`.
2. Locate the element `linguistic_processing/properties/property` and set the value of `normalize_form` to false:

```
<property name="normalize_form">false</property>
```

For example, the following setting denotes that diacritics will not be normalized:

```
<linguistic_processor>
<name>RLP</name>
<type>native</type>
<lib_path>:/xPlore/dsearch/cps/cps_daemon/bin/linguistic_processor.dll</lib_path>
<properties>
<property name="max_data_per_process">31457280</property>
<property name="config_file">
e:/xPlore/dsearch/cps/cps_daemon/cps_context.xml</property>
<property name="normalize_form">false</property>
<property name="skip_components_lang_list">de</property>
<property name="force_tokenize_on_whitespace">true</property>
<property name="keep_accents_in_original_form">true</property>
</properties>
<languages>en,en_uc,zh,zh_sc,zh_tc,ja,ko,de,fr,ru,ar,nl,cs,hu,es,it,pt</languages>
</linguistic_processor>
```

*To store original forms for words with accents:* This is very useful when you perform wildcard queries with terms using accents.

Prerequisite: The parameter `normalize_form` must be set to true.

1. Edit the CPS configuration file `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` located in `xplore_home/dsearch/cps/cps_daemon`.
2. Locate the element `linguistic_processing/properties/property` and set the value of `keep_accents_in_original_form` to true:

```
<property name="keep_accents_in_original_form">true</property>
```

If objects were already indexed, reindex them.

## Characters that are treated as white space

The default special characters are defined in `indexserverconfig.xml` as the value of the `special-characters` attribute on the `content-processing-services` element:

```
@#$%^~`&.:()_-+=<>'`\\[]{}{}
```



**Note:** The special characters list must contain only Unicode characters (first 65,536 code points).

For example, a phrase `extract-text` is tokenized as `extract` and `text`, and a search for either term finds the document.

## Characters that are required for context (punctuation)

The default context characters are defined in indexserverconfig.xml as the value of the *context-characters* attribute of the *content-processing-services* element:

```
!, ;?"
```

 **Note:** The context characters list must contain only Unicode characters (first 65,536 code points).

White space is substituted after the parts of speech have been identified. For example, the phrase Is John Smith working for OpenText? the question mark is filtered out because it functions as a context special character (punctuation).

## Special characters in queries

When a string containing a special character is indexed, the tokens are stored next to each other in the index. A search for the string is treated as a phrase search. For example, an index of *home\_base* stores *home* and *base* next to each other. A search for *home\_base* finds the containing document but does not find other documents containing *home* or *base* or *home base* or *home-base* or *home?base*.

If a query fails, check to see whether it contains a special character.

 **Note:** Reindex your documents after you change the special characters list.

## 5.12 Configuring stop words

To prevent searches on common words, you can configure stop words that are filtered out of queries. Stop words are not removed during indexing, to support phrase searches. Exceptions:

- Stop words in phrase searches are not filtered if exact-phrase-match is enabled.
- Stop words that appear within the context of context characters are not filtered. For example, if the query term is *stop?and?go*, *and* is a stop word. The question mark is defined in indexserverconfig.xml as a context character. CPS does not filter the question mark.

The following stop word lists are provided for Chinese, Korean, and Japanese in subdirectories of *xplore\_home\dssearch\cps\cps\_daemon\shared\_libraries\rlp\*. Edit them in UTF-8 encoding. Each line represents on lexeme (stop word). Blank lines or comments beginning with # are ignored.

- Chinese: cma/dicts/zh\_stop.utf8
- Korean: kma/dicts/kr\_stop.utf8
- Japanese: jma/dicts/JP\_stop.utf8

Stop word filtering is not configurable from Documentum interfaces. Stop words are always filtered for single/multi-term searches, and stop words are not filtered in phrase searches when query-exact-phrase-match is enabled.

Stop word filtering is configurable in an XQuery expression. A search with the constraint

```
(. ftcontains "the quick brown fox jumps over the lazy dog" with stemming using stop words default)
```

will filter stop words in the phrase by default. But if query-exact-phrase-match is enabled, then the stop words will be kept. When stop words are filtered, there will be a position gap to perform the query, which means the query brown fox jumps over the lazy dog is the same as query brown fox jump xxx xxx lazy dog. Therefore, jump and lazy must have position difference of 2.

A query with the constraint

```
(. ftcontains "quick brown fox jumps lazy dog" without stemming)
```

returns only documents containing the string quick brown fox jumps lazy dog and does not filter stop words.

## Enabling and adding stop word lists to xPlore

Out-of-the-box stop words lists are provided with CPS for some common languages such as English, Spanish, and French, but they are disabled by default.

To enable an out-of-the-box stop words list for a common language, edit the file `stop-options.xml` in `xplore_home\dsearch\cps\cps_daemon\shared_libraries\rlp\etc` and uncomment the line containing the dictionary path for that language, which is commented out by default. For example, uncommenting the following line turns on the stop words list for German:

```
<dictionarypath language="deu"><env name="root"/>/etc/german-stopwords.txt</dictionarypath>
```

Directly edit an out-of-the-box stop words list file (`<language>-stopwords.txt`) to configure the stop words for the corresponding language.

To add stop words lists for other languages, register your lists in the file `stop-options.xml` in `xplore_home\dsearch\cps\cps_daemon\shared_libraries\rlp\etc`. The stop words file must contain one word per line, in UFT-8 format.

The following example adds a stop words list in Spanish after the English list:

```
<dictionarypath language="eng">
  <env name="root"/>/etc/en-stopwords.txt</dictionarypath>
<dictionarypath language="es">
  <env name="root"/>/etc/es-stopwords.txt</dictionarypath>
```

A sample Spanish stopwords file:

```
a
adonde
```

```
a1
como
con
conmigo...
```

## 5.13 Searching Chinese, Japanese, or Korean Terms

Terms in Chinese, Japanese, and Korean Languages are not split with white spaces like English. To improve the search quality of these languages, CPS splits terms into single characters during the linguistic analysis. During the index phase, single characters can be configured to add to the component list for each token.

Take the phrase 中口人民共和国 as an example. Three component lists are generated for this phrase. Every list is composed of two or more single characters as shown below:

中口[中, 口], 人民[人, 民], 共和国[共, 和, 国] This feature makes documents containing Chinese, Japanese, or Korean terms more likely to be hit. Moreover, any adjacent characters are searchable no matter of whether the characters are located in the same component list or not.

For example, a query for 口人 also returns documents containing the phrase 中口人民共和国.

We recommend that you configure a higher score for original forms by increasing the value of *query-original-term-weight*, which makes documents containing original tokens more likely to be hit.

To enable this feature, from xPlore administrator set *generate\_cjk\_components* to *true* in the CPS configuration file. Additionally, the language must be added to *query-components-from-compound-words-lang-list* from xPlore administrator.



**Note:** This feature requires more disk space to store index data as more components are generated.

## 5.14 Troubleshooting content processing

### 5.14.1 CPS troubleshooting methods

#### CPS log files

If CPS is installed as an in-process service on an xPlore instance, it shares the logging for the primary instance web application. The log files *cps.log* and *cps\_daemon.log* are located in *xplore\_home/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/logs*.

#### Logging in a standalone instance

The logging configuration file is located in the CPS war file, in WEB-INF/classes. The log files *cps.log* and *cps\_daemon.log* are located in *cps\_home/<wildfly\_version>/server/DctmServer\_RemoteCPSInstanceName/logs*.

#### *Logging separate instances*

Make sure that each CPS instance specifies a different log file path. The log output file is specified in the logback.xml file of the instance. If CPS is installed as a standalone service, the logback file is located in the CPS war file, in WEB-INF/classes. If CPS is installed as an in-process service on an xPlore instance, it shares the logback file of the indexserver web application.

#### *Log levels*

In order of decreasing amount of information logged: trace, debug, info, warn, and error. Set the log level to INFO to troubleshoot CPS.

#### *Log output*

Each CPS request is logged with the prefix DAEMON#. You see prefixes for following libraries in CPS logging:

- CPS daemon: CORE
- Text extraction: TE\_STELLENT
- HTTP content retrieval: CF\_HTTP
- Language processing: LP
- Language identification: LI\_RLI

Following is an example from cps.daemon.log. (Remote CPS log is named cps\_manager.log):

```
2012-03-28 10:28:41,828  INFO [Daemon0(4496)-TE-Stellent-(136)]  
identify_file_normally_configured: true, actual: true  
...  
2012-03-28 11:16:37,204  WARN [Daemon0(4496)-Core-(3448)] LP of lpReq 0  
of sub-request 4 of req 32 of doc 080023a38000151f based on fallback lang en,  
encoding utf-16le  
2012-03-28 11:16:36,829  WARN [Daemon0(4496)-LI-RLI-(5464)] No language matched.
```

## Testing tokenization

Test the tokenization of a word or phrase to see what is indexed. Expand **Diagnostic and Utilities** in the xPlore administrator tree and then choose **Test tokenization**.

Input the text and select the language. Different tokenization rules are applied for each language. (Only languages that have been tested are listed. See the release notes for supported languages. Other languages are not tokenized.)

Uppercase characters are rendered as lowercase. White space replaces special characters.

The results table displays the original input words. The root form is the token used for the index. The Start and End offsets display the position in raw input.

Components are displayed for languages that support component decomposition, such as German.

Results can differ from tokenization of a full document for the following reasons:

- The document language that is identified during indexing does not match the language that is identified from the test.
- The context of the indexed document does not match the context of the text.

Use the executable CASample in *xplore\_home/dsearch/cps/cps\_daemon/shared\_libraries/stellent\_text\_extractor* to test the processing of a file. Syntax:

```
casample path_to_input_file
```

## 5.14.2 CPS startup and connection errors

### CPS does not start

The most common cause of CPS non-start is that the port is occupied. The error message is Failed to create server socket; the port may be occupied. The port is not tested when you configure CPS. The CPS process (CPSDaemon) runs on port 9322 by default or <*xplore\_base\_port*>+22. Make sure that the process has started. If it has not started, check the cps\_daemon log in *xplore\_home/<wildfly\_version>/server/DctmServer\_NodeInstanceName*(or *DctmServer\_RemoteCPSInstanceName*)/logs.

Go to *xplore\_home/dsearch/cps/cps\_daemon/bin* and try to run the daemon directly to see whether it can be started using the following command:

```
bin>CPSDaemon .../xxxxx_configuration.xml daemon0 9322 normal
```

Another possible cause is an unsupported OS. Verify the supported OS for your version of xPlore.

If CPS fails to start, the CPS configuration may be invalid. Check to see whether you have changed the file *NodeInstanceName\_local\_configuration.xml* or *RemoteCPSInstanceName\_configuration.xml* in *xplore\_home/dsearch/cps/cps\_daemon*.

### CPS restarts often

The CPS load can be too high, causing out of memory errors. Check the use of memory on the CPS instance.

Suggested workarounds:

- Change the CPS configuration: Decrease the number of worker threads. Increase *daemon\_count* if the host has enough CPU and memory to reduce the load for one CPSDaemon.
- Resubmit any failed files using the Documentum index agent.
- Install a standalone CPS instance, temporarily for high ingestion periods, or permanently.

## Remote CPS issues

If you try to rebuild an index without configuration, you see an error in the remote CPS daemon log like the following:

```
2011-02-16 22:53:32,646 ERROR [Daemon-Core-(3520)] ...
err-code 513, err-msg C:/xPlore/dsearch/cps/cps_daemon/export/080004578000
0e5bdmftcontentref1671968925161715182.txt cannot be opened.
The system cannot find the file specified.
```

This failure occurs because the rebuilding index module dumps large texts to the first CPS export\_path folder. Therefore, if there are multiple CPS instances which can be accessed by the xPlore instance, make sure all those CPS instances can access the first CPS' export\_path, or set up a shared folder in rebuild-index-temp-path in indexserverconfig.xml

## Communication error occurred while processing request

Resubmit the indicated document using the index agent UI. If you have more than one document with this error, resubmit them in a batch.

## Request marshal/unmarshal error

This error is caused by incompatible CPS installations. Make sure that you have upgraded all instances after you upgrade the primary xPlore instance. See *OpenText Documentum xPlore Installation Guide* for details.

### 5.14.3 Adding CPS daemons for ingestion or query processing

You can configure multiple CPS daemons for reindexing or heavy ingestion loads or add CPS daemons to increase query processing. Multiple daemons do not consume as much memory as multiple CPS instances. They have almost the same throughput as multiple CPS instances on the same memory-provisioned host.

From the xPlore administrator, do the following for each CPS instance:

1. Edit the CPS configuration file in the CPS host directory *xplore\_home/dsearch/cps/cps\_daemon*.
2. Change the value of element *daemon\_count* to 3 or more (default: 1, maximum 6).
3. Change the value of *connection\_pool\_size* to 3 or more.
4. Restart all xPlore instances.
5. (Optional for temporary ingestion loads) Reset the CPS *daemon\_count* to 1 and *connection\_pool\_size* to 5 after reindexing is complete.

## 5.14.4 Running out of space

Indexing fails when there is insufficient temp space. Both the Lucene index and the CPS daemon use temp space. Other applications (not xPlore) can also be using the temp space. A message in the index agent log indicates the problem:

```
IO_ERROR: error during add entry... Original message: No space left on device...
```

A similar error is Failed to create temporary file or error code 47 (file write error).

Ensure that the directory for the CPS temporary file path is large enough. Set the value of temp\_file\_folder in the file NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml. This file is located in *xplore\_home/dsearch/cps/cps\_daemon*. Its size should not be less than 20 GB for file processing.

### Adding temp space for CPS

Increase the space or change the location in the CPS configuration file *NodeInstanceName\_local\_configuration.xml* or *RemoteCPSInstanceName\_configuration.xml*, which is located in the CPS instance directory *xplore\_home/dsearch/cps/cps\_daemon*. After you change the temp space location, reindex or resubmit the documents that failed indexing.

## 5.14.5 CPS file processing errors

### Unsupported format or file encoding

If the CPS analyzer cannot identify the file type, it displays the following error. The XML element that contains the error is displayed:

```
*** Error: no filter available for this file type in element_name.
```

Check the file using the casample utility to see if it is recognized. See “[CPS troubleshooting methods](#)” on page 143. If the file is XML, check to see that it is well-formed. For a list of supported formats, see Oracle Outside In documentation.

If the document uses an unsupported encoding, a 1027 error code is displayed. For supported encodings, see Basitech documentation.

For the error message Unknown language provided, check to see whether you have configured an invalid locale. See “[Configuring languages and encoding](#)” on page 126.

### Empty or very small file

If the file is empty, the following error is displayed. The XML element that contains the error is displayed:

```
*** Error: file is empty in element_name.
```

If the error message is Not enough data to process, the file has very little text content and the language was not detected.

## File could not be opened

Check the CPS instance configuration for export file path.

## File corrupted

If there are processing errors for the file, they will be displayed after the processing statistics. A corrupt file returns the following error. The XML element that contains the error is displayed:

```
*** Error: file is corrupt in element_name.
```

A file with bad content can also return the error message Served data invalid.

Check the non XML file using the casample utility to see if it is corrupted. See “[CPS troubleshooting methods](#)” on page 143.

## Some documents are not indexed

- Is the collection set to read-only? Documents submitted for updating will not be indexed.
- Is the format supported by CPS? Unsupported format is the most common error. Check the list of supported formats in Oracle Outside In documentation. Check an individual file by submitting it to a test. Refer troubleshooting section about how to run casample.
- Is indexing enabled for the object type? Documents are not indexed if the document type is not registered or is not a subtype of a registered type. Check whether indexing is enabled (the type is a subtype of a registered type). You can check whether indexing is enabled in Documentum Administrator by viewing the type properties. You can get a listing of all registered types using the following iAPI command:

```
? ,c,select distinct t.name, t.r_object_id from dm_type t, dmi_registry i  
where t.r_object_id = i.registered_id
```

You see results like the following:

```
name r_object_id  
-----  
dm_group 0305401580000104  
dm_acl 0305401580000101  
dm_sysobject 0305401580000105
```

- You can register or unregister a type through Documentum Administrator. The type must be *dm\_sysobject* or a subtype of it. If a supertype is registered for indexing, the system displays the **Enable Indexing** checkbox selected but disabled. You cannot clear the checkbox.
- Is the format indexable? Check the *class* attribute of the document format. See “[Documentum attributes that control indexing](#)” on page 74 for more information.
- Is the document too large? See “[Maximum document and text size](#)” on page 124.

## Contents in input form fields are not indexed

Is the `get_text_from_vectorsavetag` property set to `true`? Contents in input form fields are not indexed if this property is not explicitly set to `true`.

```
<text_extractor>
  <name>stellent</name>
  .....
  <properties>
  .....
    <!-- whether get text from vectorsavetag-->
    <property name="get_text_from_vectorsavetag">true</property>
  </properties>
  .....
</text_extractor>
```

## 5.14.6 Troubleshooting slow ingestion and timeouts

Slow ingestion is most often seen during migration. If migration is spread over days, for example, tens of millions of documents ingested over two weeks, slow ingestion is usually not an issue. Most ingestion issues can be resolved with planning, pre-production sizing, and benchmarking.

Generally, tuning ingestion may be a time consuming procedure, you may check indexagent ingestion data and xPlore ingestion report to see which component is a bottleneck. You may also verify xPlore resource consumption report to understand which resource is a bottleneck.

### Request processing timeout

Check the timeout threshold in CPS. Try increasing the `request_time_out` parameter and resubmitting the document using the index agent UI. Check the document size and make sure that the file size and text content do not exceed the limits in index agent and CPS configuration. See “Large documents” below.

For the error messages

`File path or URI is not valid`

or

`No permission to read the file`

, check the export file path in xPlore administrator CPS instance configuration. If CPS cannot access it, it cannot retrieve the document content.

### Insufficient CPU

Content extraction and text analysis are CPU-intensive. CPU is consumed for each document creation, update, or change in metadata. Check CPU consumption during ingestion.

Suggested workarounds: For migration, add temporary CPU capacity. For day-forward (ongoing) ingestion, add permanent CPU or new remote CPS instance on other hosts. CPS instances are used in a round-robin order.

## Insufficient memory

When xPlore indexes large documents, it loads much content which consumes too much memory. In this case, you get an out-of-memory error in dsearch.log such as:

```
Internal server error. [Java heap space] java.lang.OutOfMemoryError
```

Suggested workarounds:

- Add more memory to xPlore.
- Limit the document and text size as described in “[Maximum document and text size](#)” on page 124.

## Large documents

Large documents can tie up a slow network. These documents also contain more text to process. Use xPlore administrator reports to see the average size of documents and indexing latency and throughput. The average processing latency is the average number of seconds between the time the request is created in the indexing client and the time xPlore receives the same request. The State of repository report in Documentum Server also reports document size. For example, the Documents ingested per hour reports shows number of documents and text bytes ingested. Divide bytes ingested by document count to get average number of bytes per document processed.

Several configuration properties affect the size of documents that are indexed and consequently the ingestion performance. “[Maximum document and text size](#)” on page 124 describes these settings.

## Disk I/O issues

You can detect disk I/O issues by looking at CPU utilization. Low CPU utilization and high I/O response time for ingestion or query indicate an I/O problem. Test the network by transferring large files or using Linux dd (disk dump). You can also measure I/O performance on Linux using the bonnie benchmark. On Windows, you can use sqlio.

If a query is very slow the first time and much faster the second time, you could have an I/O problem. If it is slow the second time, this performance is probably not due to insufficient I/O capacity.

*Workaround:*

- NAS: Verify that the network has not been set as half duplex. Increase network bandwidth and/or improved network I/O controllers on the xPlore host. Ensure the NAS storage is optimized for xPlore access. If it is shared by many applications, when xPlore starts a new ingestion procedure, performance may be reduced by tasks from other activities.
- SAN (check in the following order)
  1. Verify that the SAN has sufficient memory to handle the I/O rate.

2. Increase the number of drives available for the xPlore instance.
3. If the SAN is multiplexing a set of drives over multiple application, move the disk space to a less contentious set of drives.
4. If other measures have not resolve the problem, change underlying drives to solid state.
5. Use striped mapping instead of concatenated mapping so that all drives can be used to service I/O.

## Slow network

A slow network between the Documentum Server and xPlore results in low CPU consumption on the xPlore host. Consumption is low even when the disk subsystem has a high capacity. File transfers via FTP or network share are also slow, independent of xPlore operations.

*Workaround:* Verify that network is not set to half duplex. Check for faulty hubs or switches. Increase network capacity.

## Large number of Excel documents

Microsoft Excel documents require the most processing of all text formats, due to the complexity of extracting text from the spreadsheet structure. You can detect the number of Excel documents using the State of repository report in Documentum Server.

Suggested workaround: Add temporary CPU for migration or permanent CPU for ongoing load.

## Virus checking software

Virus checking software can lead to high disk I/O because it continually checks the changes in xPlore file structures during indexing.

Workarounds: Exclude temp and xPlore working and data directories, or switch to Linux platform.

## Interference by another guest OS

In a VM environment, the physical host can have several guest operating systems. This contention could cause intermittent slowness in indexing unrelated to format, document size, I/O capacity, or CPU capacity.

*Workaround:* Consult with your infrastructure team to load balance the VMs appropriately.

## Slow content storage area

Ingestion is dependent on the speed of the content source. Content storage issues are especially noticeable during migration. For example, you find that migration or

ingestion takes much longer in production than in development. Development is on a small volume of content on NAS but production content is on a higher-latency device like Centera. You can determine the location of the original content by using the State of the repository report in Documentum Server.

*Workaround:* Extend the migration time window.

### Concurrent large file ingestion

When CPS processes two or more large files at the same time, the CPS log file reports one of the following errors (`cps_daemon.log`):

```
2012-03-28 10:28:41,828
ERROR [Daemon-Core-(3400)] Exception happened, ACCESS_VIOLATION,
Attempt to read data at address 1 at (connection-handler-2)
...
FATAL [DAEMON-LP_RLP-(3440)] Not enough memory to process linguistic requests.
Error message: bad allocation
```

Use xPlore administrator to select the instance, and then choose **Configuration**. Change the following to smaller values:

- Max text threshold
- Thread pool size

You can add a separate CPS instance that is dedicated to processing. This processor does not interfere with query processing.

## 5.15 Adding dictionaries to CPS

You can create user dictionaries for words specific to an industry or application, including personal names and foreign words. In your custom dictionary, you specify the part of speech for ambiguous nouns or verbs, which assists CPS in determining the context for a sentence. You can also prevent words from being decomposed, making queries more precise.

The following procedure creates a Chinese user dictionary. Use these same steps for other supported languages (Japanese and Korean)

1. Create a UTF-8 encoded file. Each entry in the file is on a single line with the following syntax (TAB-delimited):

word	part_of_speech	decomposition_pattern
------	----------------	-----------------------

part\_of\_speech: NOUN, PROPER\_NOUN, PLACE, PERSON, ORGANIZATION, GIVEN\_NAME, or FOREIGN\_PERSON.

decomposition\_pattern: A comma-delimited list of numbers that specify the number of characters from *word* to include in each part of the compound. A value of 0 indicates no decomposition. For example, the following entry indicates that the word is decomposed into three two-character sequences. The sum of the digits in the pattern must match the number of characters in the entry.

**深圳发展銀行 organization 2,2,2**

The following example is decomposed into two four-character sequences:

**深圳发展銀行 noun 2,4**

2. Compile the dictionary.

On both Linux and Windows, there is a compilation script for each supported language. Use the one that corresponds to the language of your dictionary:

Chinese: <xplore\_home>\dsearch\cps\cps\_daemon\shared\_libraries\rlp\cma\source\samples\build\_user\_dict.sh

Japanese: <xplore\_home>\dsearch\cps\cps\_daemon\shared\_libraries\rlp\jma\source\samples\build\_user\_dict.sh

Korean: <xplore\_home>\dsearch\cps\cps\_daemon\shared\_libraries\rlp\kma\source\samples\build\_user\_dict.sh

- On Linux:

1. Export the following variables:

```
export BT_ROOT=<xplore_home>/dsearch/cps/cps_daemon/shared_
libraries
export BT_BUILD=<variableDir>
```

Where <variableDir> is a subdirectory under <xplore\_home>/dsearch/cps/cps\_daemon/shared\_libraries/rlp/bin/ and its name differs from computer to computer. For example:

```
export BT_BUILD=amd64-glibc25-gcc42
```

```
export LD_LIBRARY_PATH=<libbtutils.so.7.2_path>
```

Where <libbtutils.so.7.2\_path> is the path of the libbtutils.so.7.2 library. For example:

```
export LD_LIBRARY_PATH=/opt/dmadmin/xPlore1.5/dsearch/cps/cps_daemon/shared_libraries/rlp/lib/amd64-glibc25-gcc42
```

2. Use the chmod command to change the file permissions on the compilation script and some other files:

```
chmod a+x build_user_dict.sh
chmod a+x build_cla_user_dictionary
chmod a+x cla_user_dictionary_util
chmod a+x t5build
chmod a+x t5sort
```

3. Run the compilation script build\_user\_dict.sh. Use the following as an example:

```
./build_user_dict.sh mydict.txt mydict.bin
```

- On Windows:

1. Download and install Cygwin from <http://www.cygwin.com/>.

2. Launch the Cygwin terminal.
3. Use the chmod command to change the file permissions on the compilation script: chmod a+x build\_user\_dict.sh
4. Export the following variables:

```
export BT_ROOT=<xplore_home>/dsearch/cps/cps_daemon/shared_
libraries
export BT_BUILD=<variableDir>
```

Where <variableDir> is a subdirectory under <xplore\_home>/dsearch/cps/cps\_daemon/shared\_libraries/rlp/bin/ and its name differs from computer to computer. For example:

```
export BT_BUILD=amd64-w64-msvc100
```

5. Edit the build\_user\_dict.sh file to make sure that carriage returns are denoted by \n instead of \n\r in the file.
6. In the Cygwin terminal, run the compilation script build\_user\_dict.sh specific to the dictionary language; for example:

```
./build_user_dict.sh mydict.txt mydict.bin
```

3. Put the compiled dictionary into the directory specific to the dictionary language:
  - Chinese: xplore\_home/cps/cps\_daemon/shared\_libraries/rlp/cma/dicts
  - Japanese: xplore\_home/cps/cps\_daemon/shared\_libraries/rlp/jma/dicts
  - Korean: xplore\_home/cps/cps\_daemon/shared\_libraries/rlp/kma/dicts
4. Edit the CLA configuration file to include the user dictionary. Add a *dictionarypath* element to cla-options.xml in xplore\_home/cps/cps\_daemon/shared\_libraries/rlp/etc.

The following example adds a Chinese user dictionary named mydict.bin:

```
<claconfig>
  ...
  ...
  <dictionarypath><env name="root" />/cma/dicts/mydict.bin
  </dictionarypath>
</claconfig>
```

5. To prevent a word that is also listed in a system dictionary from being decomposed, modify cps\_context.xml in xplore\_home/cps/cps\_daemon. Add the property *com.basistech.cla.favor\_user\_dictionary* if it does not exist, and set it to true.

For example:

```
<contextconfig><properties>
  <property name="com.basistech.cla.favor_user_dictionary"
    value="true" />...
```

6. Restart CPS for the changes to take effect.

## 5.16 Custom content processing

### 5.16.1 About custom content processing

CPS uses embedded content processors for text extraction, language identification, and linguistic analysis. These processes are thoroughly tested and supported, and they are adequate for most content processing needs. You can add custom content processors to address the following use cases:

- Custom text extraction that processes certain mime-types.
- Normalization: Normalizing varied inputs of data like phone numbers or customer IDs. For example, customer IDs can be stored as 123-abc-456-789 but only the 456-789 digits are significant. Normalization would extract this from the text so that users would find the document when they search for “456789” or “456-789”.
- Entity extraction and XML annotation: For example, extracting the locations of people or places from the full-text content.

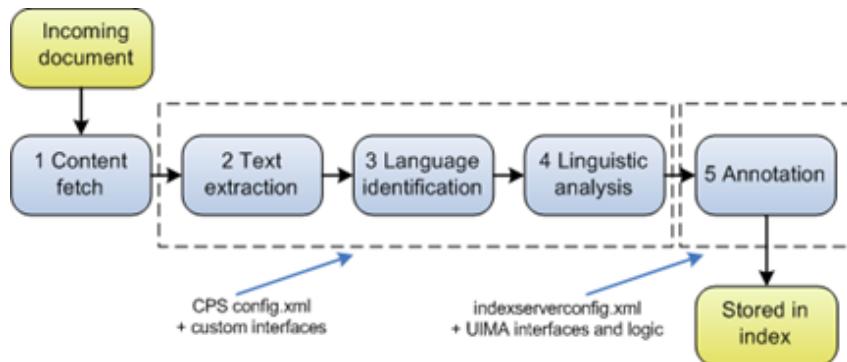
You can customize the CPS document processing pipeline at the following stages:

1. Text extractor for certain content formats, both Java and C/C++ based. Supports custom file decryption before extraction. Specified in CPS  
NodeInstanceName\_local\_configuration.xml or  
RemoteCPSInstanceName\_configuration.xml.

Custom text extractors are usually third-party modules that you configure as text extractors for certain formats (mime types). You must create adaptor code based on proprietary, public xPlore adaptor interfaces.

2. Annotators: Classify elements in the text, annotate metadata, and perform name indexing for faster retrieval.

Custom annotators require software development of modules in the Apache UIMA framework.



**Figure 5-3: Custom content processing**

## Support for plugins

OpenText supports xPlore default text extraction, language identification, and linguistic analysis. A few custom plugins have been tested to validate the plugin environment, but these plugins themselves are not supported. The plugin runtime is sandboxed, and plugin issues are reported separately. OpenText is not responsible for the performance, memory consumption, and stability of your plugins and annotators.

Sample Java and C++ text extraction plugins and readmes are provided in `xplore_home/dsearch/cps/cps_daemon/sdk` and `cps/cps_manager`.

For best troubleshooting of your plugins, configure your custom component on a separate CPS instance. Documents that meet criteria in your plugin or UIMA module are routed to this CPS instance. Ensure that you have deployed on the host any libraries, such as MSVC 2008 distribution, that your plugins are dependent on.

## Customization steps

To customize a plugin, do the following:

1. Write plugin (Java or C/C++) or UIMA annotator.
2. Place DLLs or jar files in the CPS classpath.
3. Modify configuration from xPlore administrator.
4. Repeat for each CPS instance.
5. Test content processing.
6. Perform a backup of your customization DLLs or jars when you back up the xPlore federation.

## Configuration updates

To modify plugin configurations across all CPS instances, do the following:

1. In Services > Content Processing Service, click **CPS Plugin Configuration**.
  2. Modify configuration settings and save changes.
- The changes are applied to all CPS instances.



**Note:** The following tips will help prevent issues related to configuration changes:

- To avoid re-indexing the docbase or rebuilding the index, ensure there is no data in the system when changing configurations for text extraction and linguistic processing.
- Do not change configurations when indexing and search are being executed concurrently.

- Changes to default configurations must be reapplied when a new CPS/node is added.

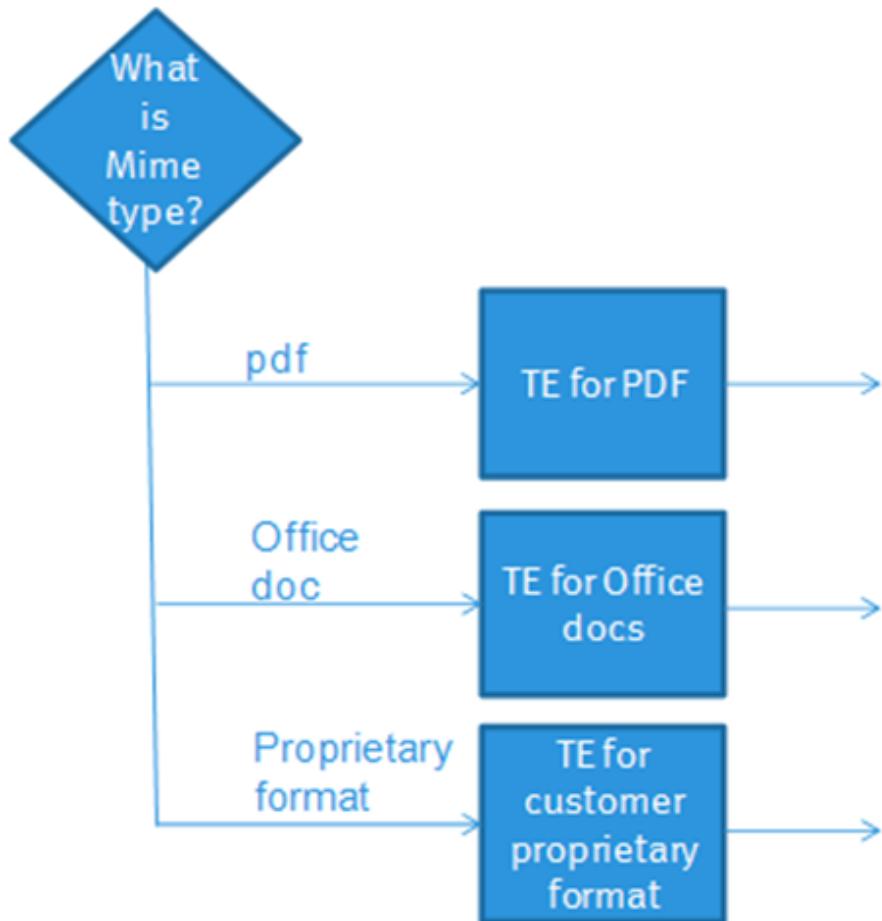
## 5.16.2 Text extraction

The text extraction phase of CPS can be customized at the following points:

- Pre-processing plugin
- Plugins for text extraction based on mime type, for example, the xPlore default extractor Oracle Outside In (formerly Stellent) or Apache Tika.
- Post-processing plugin

For best reliability, deploy a custom text extractor on a separate CPS instance. For instructions on configuring a remote CPS instance for text extraction, see “[Adding a remote CPS instance](#)” on page 120.

The following diagram shows three different mime types processed by different plugins

**Figure 5-4: Custom text extraction based on mime type**

### Configuring a text extraction plugin

Add your plugins to `NodeInstanceName_local_configuration.xml` or `RemoteCPSInstanceName_configuration.xml` in `xplore_home/dsearch/cps/cps_daemon`. Each extractor requires the following information:

**Table 5-2: Child elements of text\_extraction**

Element	Description
<code>text_extractor_preprocessor</code>	Preprocessing specification, contains name, type, lib_path, formats, properties
<code>text_extractor</code>	Contains name, type, lib_path, formats, properties
<code>text_extractor_postprocessor</code>	Postprocessing specification, contains name, type, lib_path, formats, properties

Element	Description
name	Used for logging
type	Valid values: Java or native (C/C++)
lib_path	Fully qualified class name in CPS host classpath.
formats	Contains one or more format element. Each format element corresponds to a mime type.
properties	Contains user-defined property elements.
property	The value of each named property element is read by your plugin.

## Creating a text extractor adaptor class

xPlore provides public, proprietary interfaces that you can implement for your plugin. Implement the abstract class CPSTextExtractor in the package com.emc.cma.cps.processor.textextractor.

## Sample Tika text extractor

The Apache Tika toolkit extracts metadata and structured text from documents. Sample jar files and Tika configuration file are installed in *cps\_home/dsearch/cps/add-ons/tika*. Other samples are in *cps\_home/dsearch/cps/cps\_daemon/sdk*.

1. Download and place Tika jar files in *xplore\_home/dsearch/cps/add-ons/tika*.
2. Back up NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml in *xplore\_home/dsearch/cps/cps\_daemon*.
3. Copy configuration\_tika.xml, located in *xplore\_home/dsearch/cps/add-ons/tika*, to *xplore\_home/dsearch/cps/cps\_daemon*.
4. Rename the copied file to NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml in step 2.

The following example contains a preprocessor and an extractor plugin. The properties are passed to the plugin for processing. In this example, return\_attribute\_name is a property required by the Tika text extractor.

```
<cps_pipeline>
  ...
<text_extraction>
  <!--preprocessor chain invoked in the configured sequence-->
  <text_extractor_preprocessor>
    <name>password_cracker</name>
    <type>java</type>
    <lib_path>
      com.emc.cma.cps.processor.textextractor.CPSPasswordCracker
    </lib_path>
    <properties>
      <property name="return_attribute_name">false</property>
    </properties>
    <formats>
```

```
    <format>application/pdf</format>
  </formats>
</text_extractor_preprocessor>

<text_extraction>
<text_extractor>
  <name>tika</name>
  <type>java</type>
  <lib_path>
    com.emc.cma.cps.processor.textextractor.CPSTikaTextExtractor
  </lib_path>

  <properties>
    <property name="return_attribute_name">false</property>
  </properties>
  <formats>
    <format>application/pdf</format>
  </formats>
</text_extractor> ...
</text_extraction>
...
</cps_pipeline>
```

### 5.16.3 Troubleshooting custom text extraction

Set up custom text extraction (TE) on a remote CPS instance. Configure the instance to do text extraction only.

Custom text extraction plugins raise the following errors:

- Missing or misplaced custom library. The CPS daemon does not start, and `cps_daemon.log` reports an initialization error.
- A file with the incorrect mime type is routed to the TE plugin. Change the default log level from `WARN` to `DEBUG` and ingest the same document. If two text extractors are available for a given format, the first one in the configuration receives the request.
- If the text extractor hangs during daemon processing, the CPS daemon log records a timeout error and the daemon restarts. If the text extractor hangs on the manager side, processing is blocked and requires a CPS restart.
- If the text extractor crashes during daemon processing, the CPS daemon log restarts. If the text extractor crashes on the manager side, processing is blocked and requires a CPS restart.

### 5.16.4 Annotation

Documents from a Documentum Server are submitted to CPS as `dftxml` files. The CPS annotation framework analyzes the `dftxml` content. Text can be extracted to customer-defined categories, and the metadata can be annotated with information.

Annotation employs the Apache UIMA framework. A UIMA annotator module extracts data from the content, optionally adds information, and puts it into a consumable XML output.

A UIMA annotator module has the following content:

- An XML annotation descriptor file.
- A common analysis structure (CAS) that holds the input and output for each stage in the sequence. The type system descriptor file, also known as a consumer descriptor, defines the type structure for the CAS. The CAS tool generates supporting code from the type descriptor file.
- Annotation implementation code that extends JCasAnnotator\_ImplBase in the UIMA framework.
- An analysis engine consisting of one or more annotators in a sequence.

Sample annotation module files are provided in the xPlore SDK:

- Sample annotator module (`dsearch-uima.jar`) in `/samples/dist`.
- Annotator source files in `/samples/src/com/emc/documentum/core/fulltext/indexserver/uima/ae`.
- Sample `indexserverconfig.xml` for the example in `/samples/config`
- Sample type descriptor and annotation definition files in `/samples/src/com/emc/documentum/core/fulltext/indexserver/uima/descriptors/`

## Steps in creating a UIMA annotator for xPlore

Following is a high-level description of the steps. See [the example](#) for implementation details.

1. Download the Apache UIMA Java framework and SDK binaries and add them to your project build path.
2. Create a CAS results type definition file that specifies a feature element for each data type that you are annotating.
3. Use the Apache UIMA tool `jcasgen.bat` (Windows) or `jcasgen.sh` (Linux) to generate type implementation classes from the type definition file.
4. Create the annotator descriptor XML file that describes the annotator detail. The descriptor describes the name of the annotator, the output type, and the runtime configuration parameters.
5. Create annotator code that extends `JCasAnnotator_ImplBase` in the UIMA framework.
6. Compile the Java classes and package with the XML descriptor files in a jar file. Copy the jar file to `xplore_home/<wildfly_version>/server/DctmServer_NodeInstanceName/deployments/dsearch.war/WEB-INF/lib`.
7. Deploy to xPlore:
  - a. Stop all xPlore instances.
  - b. Specify UIMA modules and configure their usage in `indexserverconfig.xml`.

- c. Change `indexserverconfig.xml` pipeline element to specify the descriptor, process-class, and any properties for the annotator. Add pipeline-usage for the associated category of documents.
- d. Restart the xPlore instances.



**Note:** To deploy the UIMA module as a remote service, you can use the Vinci service that is included in the UIMA SDK.

## Specifying UIMA modules in `indexserverconfig.xml`

Stop all xPlore instances and add your UIMA references to `indexserverconfig.xml`, located in `xplore_home/config`. Add a `pipeline-config` element as a child of `index-server-configuration` between `content-processing-services` and `category-definitions`. The `pipeline-config` element has the following syntax:

```
<pipeline-config>
  <pipeline descriptor="UIMA-pipeline-descriptor.xml"
    process-class="xPlore_UIMAProcessFactory_class"
    name="name-of-pipeline" />
</pipeline-config>
```

**Table 5-3: Pipeline configuration attributes and elements**

Element	Description
<code>name</code>	Unique identifier of the UIMA module
<code>descriptor</code>	UIMA annotator descriptor file path. The path is resolved using the classpath.
<code>process-class</code>	The xPlore factory class. This class must implement <code>IFtProcessFactory</code> . The default class, <code>com.emc.documentum.core.fulltext.indexserver.uima.UIMAProcessFactory</code> , provides for most processing needs.
<code>properties</code>	Contains property elements that define runtime parameters.

The following example configures a UIMA module. You specify a descriptor XML file, a processing class, and an optional name. The descriptor file, and process class are hypothetical. The path to the descriptor file is from the base of the UIMA module jar file.

```
<pipeline-config>
  <pipeline descriptor="descriptors/PhoneNumberAnnotator.xml" process-class="
    com.emc.documentum.core.fulltext.indexserver.uima.UIMAProcessFactory" name="
    phonenumbers_pipeline" />
</pipeline-config>
```

The xPlore UIMAProcessFactory class is provided with xPlore. It executes the pipeline based on the definitions provided.

## Configuring UIMA module usage in indexserverconfig.xml

Configure the usage of the UIMA module within a *category* element in indexserverconfig.xml. Add one *pipeline-usage* element after the indexes element. Most applications annotate the dftxml category.

Properties control the level of concurrency and the data elements from the dftxml document that are sent to the UIMA annotator.

**Table 5-4: Pipeline usage attributes and elements**

Element	Description
name	Attribute of pipeline-usage that references a pipeline module defined in pipeline-config.
root-result-element	Attribute of pipeline-usage that specifies an element in the dftxml document that will be the root of the annotated results.
mapper-class	Optional attribute of pipeline-usage that maps the annotation result to an XML subtree, which can then be added to dftxml.
input-element	One or more optional child elements of pipeline-usage that specifies the dftxml elements that are passed to UIMA analysis engine. Use this to annotate based on a Document attribute. The name attribute must be unique within the same pipeline. The element-path attribute has an xpath value that is used to locate the XML element in the dftxml document.
type-mapping	One or more elements that specify how to store the analytic processing results in the original dftxml document. The name attribute is a unique identifier for logging. The element-name attribute specifies the XML element that holds the result value from the annotator. This element becomes a new child element within dftxml. The type-name attribute specifies the fully qualified class name for the class that renders the result. This value should match the typeDescription name in the descriptor file.
feature-mapping	One or more optional child elements of type-mapping. The required feature attribute corresponds to a data member of the type. The required element-name attribute corresponds to the XML element that the feature is mapped to.

Element	Description
properties	Child element of pipeline-usage. Specifies concurrency and the elements that are passed to the annotator.
property name= thread-count	Sets concurrency level in the underlying pipeline engine. By default, it has the same value (100) as CPS-threadpool-max-size in xPlore administrator indexing configuration.
property name= send-content-text	Controls whether to pass the content text to the annotator. Default: true. If the annotator does not require content text, set to false.
property name= send-content-tokens	Controls whether to pass the content tokens to the annotator. Default: false. If the annotator operates on tokens, set to true.

In the following example of the Apache UIMA room number example, the annotated content is placed under dmftcustom in the dftxml file (root-result-element). The content of the element r\_object\_type (input-element element-path) and object\_name are passed to the UIMA analyzer. (If you are annotating content and not metadata, you do not need input-element.) For the object\_name value, a room element is generated by the RoomNumber class. Next, the building and room-number elements are generated by a lookup of those features (data members) in the RoomNumber class.

```
<pipeline-usage root-result-element="dmftcustom" name="test_pipeline">
<input-element element-path="/dmftdoc/dmftmetadata//r_object_type" name="object_type"/>
<input-element element-path="/dmftdoc/dmftmetadata//object_name" name="object_name"/>
<type-mapping element-name="room" type-name="com.emc.documentum.core.fulltext.indexserver.uima.ae.RoomNumber">
  <feature-mapping element-name="building" feature="building"/>
  <feature-mapping element-name="room-number" feature="roomnumber"/>
</type-mapping>
<type-mapping type-name="com.emc.documentum.core.fulltext.indexserver.uima.ae.DateTimeAnnot" element-name="datetime"/>
</pipeline-usage>
```

See the Apache UIMA documentation for more information on creating the annotator class and descriptor files. See [a simple annotator example](#) for xPlore.

## UIMA in the log

If there are errors in configuration or UIMA module, you see no UIMA logging.

You can configure logback.xml with the package names of the annotator to get logs on the UIMA module. Logback.xml configuration is described in [“Configuring logging” on page 357](#).

When an annotator is initialized, you see the category in the log like the following:

```
Create Annotator 'Phone Number Annotator' from descriptor descriptors/PhoneNumberAnnotator.xml for category dftxml
```

When an annotator is applied (tracing for dsearchindex is INFO), you see the domain name and category in the log like the following:

```
Domain test category dftxml,
Apply Annotator 'Phone Number Annotator' to document
testphonenumbers5_txt1318881952107
```

## 5.16.5 Apache UIMA example

The following example is from the Apache UIMA software development kit. It is used to create a UIMA module that normalizes phone numbers for fast identification of results in xPlore.

### Prepare the environment

Download the Apache UIMA Java framework and SDK binaries and add them to your project build path.

### Define a CAS feature structure type

Define a CAS feature structure type in an XML file called a type system descriptor. This file specifies a feature element for each data type that you are annotating. The following example PhoneAnnotationTypeDef.xml creates the following string-type features. The class that handles the types is specified as the value of types/typeDescription/name: FtPhoneNumber.:

- phoneNumber: Phone number as it appears in a document.
- normalizedForm: Normalized phone number

Note that the type descriptor must import the xPlore type definition descriptors, which handle file access.

```
<?xml version="1.0" encoding="UTF-8" ?>
<typeSystemDescription xmlns="http://uima.apache.org/resourceSpecifier">
  <name>TutorialTypeSystem</name>
  <description>Phone number Type System Definition</description>
  <vendor>The Apache Software Foundation</vendor>
  <version>1.0</version>
  <imports>
    <import location="AnnotationTypeDef.xml" />
    <import location="FtDocumentAnnotationTypeDef.xml" />
  </imports>
  <types>
    <typeDescription>
      <name>FtPhoneNumber</name>
      <description></description>
      <supertypeName>uima.tcas.Annotation</supertypeName>
      <features>
        <featureDescription>
          <name>phoneNumber</name>
          <description />
          <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
        <featureDescription>
          <name>normalizedForm</name>
          <description />
          <rangeTypeName>uima.cas.String</rangeTypeName>
        </featureDescription>
      </features>
    </typeDescription>
  </types>
</typeSystemDescriptor>
```

```
</typeDescription>
</types>
</typeSystemDescription>
```

## Generate the type implementation class

Use the Apache UIMA tool jcasgen.bat (Windows) or jcasgen.sh (Linux) to generate type implementation classes from the type definition file. Eclipse and other programming environments have UIMA plugins that generate the classes. Two classes are generated:

- FtPhoneNumber
- FtPhoneNumber\_Type

You will later reference the first class in the type-mapping element of indexserverconfig.xml.

## Create an annotator descriptor

Create the annotator descriptor XML file that describes the annotator. The descriptor describes the name of the annotator, inputs and outputs as defined in the type system descriptor, and the runtime configuration parameters that the annotator accepts.

In the following example PhoneNumberAnnotator.xml, the annotator name is Phone Number Annotator. This name is used in dsearch.log to specify annotator instantiation and, when dsearchindex logging is set to *INFO*, annotator application to a document.

The configurationParameters element defines a parameter named *Patterns*. The element configurationParameterSettings defines the patterns as two regular expression patterns, each in a value/array/string element.

The typeSystemDescription element references the type system descriptor PhoneAnnotationTypeDef.xml.

The outputs are the features referenced in the same type definition. In this descriptor, the class that generates the output is referenced along with the feature. For example, the class FtPhoneNumber generates the output for both the phoneNumber and normalizedForm features.

You specify this descriptor file when you configure UIMA in xPlore. Provide the filename as the value of the descriptor attribute of the pipeline element.

```
<analysisEngineDescription xmlns="http://uima.apache.org/resourceSpecifier"
  xmlns:xi="http://www.w3.org/2001/XInclude">
  <frameworkImplementation>org.apache.uima.java</frameworkImplementation>
  <primitive>true</primitive>
  <annotatorImplementationName>PhoneNumberAnnotator</annotatorImplementationName>
  <analysisEngineMetaData>
    <name>Phone Number Annotator</name>
    <description>Searches for phone numbers in document content.</description>
    <version>1.0</version>
    <vendor>The Apache Software Foundation</vendor>
```

```

<configurationParameters>
    <configurationParameter>
        <name>Patterns</name>
        <description>Phone number regular expression patterns.</description>
        <type>String</type>
        <multiValued>true</multiValued>
        <mandatory>true</mandatory>
    </configurationParameter>
</configurationParameters>
<configurationParameterSettings>
    <nameValuePair>
        <name>Patterns</name>
        <value>
            <array>
                <string>\b+\d{3}-\d{3}-\d{4}</string>
                <string>\(\d{3}\)\s*\d{3}-\d{4}</string>
            </array>
        </value>
    </nameValuePair>
</configurationParameterSettings>
<typeSystemDescription>
    <imports>
        <import location="PhoneAnnotationTypeDef.xml"/>
    </imports>
</typeSystemDescription>
<capabilities>
    <capability>
        <inputs></inputs>
        <outputs>
            <type>FtPhoneNumber</type>
            <feature>FtPhoneNumber:phoneNumber</feature>
            <feature>FtPhoneNumber:normalizedForm</feature>
        </outputs>
        <languagesSupported></languagesSupported>
    </capability>
</capabilities>
<operationalProperties>
    <modifiesCas>true</modifiesCas>
    <multipleDeploymentAllowed>true</multipleDeploymentAllowed>
    <outputsNewCAses>false</outputsNewCAses>
</operationalProperties>
</analysisEngineMetaData>
</analysisEngineDescription>

```

## Create annotator code to generate output

The annotator is a subclass of JCasAnnotator\_ImplBase in the Apache UIMA framework. The main method to implement is the following:

```
public void process(JCas aJCas) throws AnalysisEngineProcessException
```

**Imports:**

```

import org.apache.uima.analysis_engine.AnalysisEngineProcessException;
import org.apache.uima.jcas.JCas;
import org.apache.uima.analysis_component.JCasAnnotator_ImplBase;
import org.apache.uima.UimaContext;
import org.apache.uima.resource.ResourceInitializationException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

```

**Class:**

```

public class PhoneNumberAnnotator extends JCasAnnotator_ImplBase {
    private Pattern[] mPatterns;

    public void initialize(UimaContext aContext)
        throws ResourceInitializationException {

```

```

super.initialize(aContext);
// Get config. parameter values from PhoneNumberAnnotator.xml
String[] patternStrings = (String[]) aContext.getConfigParameterValue("Patterns");

// compile regular expressions
mPatterns = new Pattern[patternStrings.length];
for (int i = 0; i < patternStrings.length; i++) {
    mPatterns[i] = Pattern.compile(patternStrings[i]);
}
}

private String normalizePhoneNumber(String input)
{
    StringBuffer buffer = new StringBuffer();
    for (int index = 0; index < input.length(); ++index)
    {
        char c = input.charAt(index);
        if (c >= '0' && c <= '9')
            buffer.append(c);
    }
    return buffer.toString();
}

public void process(JCas aJCas) throws AnalysisEngineProcessException {

    // get document text
    String docText = aJCas.getDocumentText();

    // loop over patterns
    for (int i = 0; i < mPatterns.length; i++) {
        Matcher matcher = mPatterns[i].matcher(docText);
        while (matcher.find()) {
            // found one - create annotation
            FtPhoneNumber annotation = new FtPhoneNumber(aJCas);
            annotation.setBegin(matcher.start());
            annotation.setEnd(matcher.end());
            annotation.addToIndexes();

            String text = annotation.getCoveredText();
            annotation.setPhoneNumber(text);
            annotation.setNormalizedForm(normalizePhoneNumber(text));
        }
    }
}
}

```

## Configure UIMA modules in indexserverconfig.xml

Stop all xPlore instances and add your UIMA references to indexserverconfig.xml, located in *xplore\_home/config*. Add a pipeline-config element as a child of index-server-configuration between content-processing-services and category-definitions. The xPlore UIMAProcessFactory class is provided with xPlore. It executes the pipeline based on the definitions provided.

The following example specifies the annotator descriptor file:

```

</content-processing-services>
<pipeline-config>
    <pipeline descriptor="descriptors/PhoneNumberAnnotator.xml" process-class=
        "com.emc.documentum.core.fulltext.indexserver.uima.UIMAProcessFactory" name=
        "phonenumber_pipeline"/>
</pipeline-config>

```

Configure the usage of the UIMA module within a *category* element in indexserverconfig.xml. Add one *pipeline-usage* element after the indexes element.

Most applications annotate the dftxml category, so you place pipeline-usage in category name=dftxml. For input-element, specify a name for logging and a path to the input element in dftxml. (For a sample dftxml, see “[Extensible Documentum DTD](#)” on page 413. For type-mapping, specify an element name for logging (usually the same as type-name). For type-name, specify the type definition class. For feature-mapping, specify the output XML element for element-name and the feature name that is registered in the annotator descriptor PhoneNumberAnnotator.xml. For example:

```
</indexes>
<pipeline-usage root-result-element="ae-result" name="phonenumbers_pipeline">
  <input-element name="content" element-path="/dmftdoc/dmftcontents"/>
  <type-mapping element-name="FtPhoneNumber" type-name="FtPhoneNumber">
    <feature-mapping element-name="phoneNumber" feature="phoneNumber"/>
    <feature-mapping element-name="phoneNormalized" feature="normalizedForm"/>
  </type-mapping>
</pipeline-usage>
```

## 5.16.6 Custom content processing errors

The report *Document processing error summary* retrieves errors in custom content processing. The error type is Unknown error, Corrupt file, or Password-protected or encrypted file. The name of the custom content processing pipeline is retrieved from CPS NodeInstanceName\_local\_configuration.xml or RemoteCPSInstanceName\_configuration.xml as the value of text\_extraction/name. Error Text examples:

```
Password-protected or encrypted file (native error:TIKA-198...)
Unknown error during text extraction (native error:TIKA-198...)
```

You can identify the origin of the CPS processing error in the files cps.log and dsearch.log:

- CPS log examples: Failed to extract text from password-protected files:

```
2011-08-02 23:27:23,288 ERROR
[MANAGER-CPSLinguisticProcessingRequest-(CPSWorkerThread-1)]
Failed to extract text for req 0 of doc VPNwithPassword_zip1312352841145,
err-code 770, err-msg: Corrupt file (native error:TIKA-198:
Illegal IOException from org.apache.tika.parser.pkg.PackageParser@161022a6)

2011-08-02 23:36:27,188 ERROR
[MANAGER-CPSLinguisticProcessingRequest-(CPSWorkerThread-2)]
Failed to extract text for req 0 of doc tf_protected_doc1312353385777,
err-code 770, err-msg: Corrupt file (native error:
Unexpected RuntimeException from
org.apache.tika.parser.microsoft.OfficeParser@3b11d63f)
```

- dsearch log examples: Corrupt file:

```
<event timestamp="2011-08-02 23:27:24,994" level="WARN" thread="IndexWorkerThread-1" logger="com.emc.documentum.core.fulltext.index" timeInMilliSecs="1312352844994">
<message>
<![CDATA[Document id: VPNwithPassword_zip1312352841145, message:
CPS Warning [Corrupt file (native error:TIKA-198:
Illegal IOException from
org.apache.tika.parser.pkg.PackageParser@161022a6)].]]></message>
</event>

<event timestamp="2011-08-02 23:36:28,306" level="WARN" thread="IndexWorkerThread-2" logger="
```

```
com.emc.documentum.core.fulltext.index" timeInMilliSecs="1312353388306">
<message>
<![CDATA[Document id: tf_protected_doc1312353385777, message:
CPS Warning [Corrupt file (native error:
Unexpected RuntimeException from
org.apache.tika.parser.microsoft.OfficeParser@3b11d63f)].]]></message>
</event>
```

# Chapter 6

## Indexing

### 6.1 About indexing

The indexing service receives batches of requests to index from a custom indexing client like the Documentum index agent. The index requests are passed to the content processing service, which extracts tokens for indexing and returns them to the indexing service. You can configure all indexing parameters by choosing **Global Configuration** from the **System Overview** panel in xPlore administrator.

For information on creating Documentum indexes, see “[Creating custom indexes](#)” on page 181.

Modify indexes by editing indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54. By default, Documentum content and metadata are indexed. You can tune the indexing configuration for specific needs. A full-text index can be created as a path-value index with the FULL\_TEXT option.

For information on scalability planning, see *Documentum xPlore Installation Guide*.

### 6.2 Configuring text extraction

CPS performs text extraction by getting text and metadata from binary or XML documents. CPS then transforms the results into UTF-16 half-width encoding.

Configure indexing in *xplore\_home/config/indexserverconfig.xml*. You can configure compression and how XML content is handled. Excluding content from extraction shrinks the index footprint and speeds up ingestion.

*Indexing depth:* Only the leaf (last node) text values from subelements of an XML node are returned. You can configure indexing to return all node values instead of the leaf node value. (This change negatively affects performance.) Set the value of *index-value-leaf-node-only* in the *index-plugin* element to false. Reindex your documents to see the other nodes in the index.

The paths in the configuration file are in XPath syntax and see the path within the dftxml representation of the object. (For information on dftxml, see “[Extensible Documentum DTD](#)” on page 413.) Specify an XPath value to the element whose content requires text extraction for indexing as the value of the path attribute.

**Table 6-1: Extraction configuration options**

Option	Description
do-text-extraction	Contains one or more <i>for-element-with-name</i> elements.
for-element-with-name	Specifies elements that define content or metadata that should be extracted for indexing.
for-element-with-name/xml-content	<p>When a document to be indexed contains embedded XML content, you must specify how that content should be handled:</p> <ul style="list-style-type: none"> <li>• Whether XML content can be tokenized (<i>tokenize</i>=“true   false”).</li> <li>• Whether XML content can be stored within the input document (<i>store</i>=“embed   none”).</li> </ul> <p> <b>Note:</b> When the <i>store</i> attribute is set to <i>none</i>, XML documents are not tokenized even if the <i>tokenize</i> attribute is set to <i>true</i>.</p> <ul style="list-style-type: none"> <li>• Whether XML attributes can be indexed and searched (<i>include-attribute-value</i>=“true   false”). See “<a href="#">XML attribute support</a>” on page 180.</li> </ul> <p>If your repository has many XML documents and a summary is requested with search results, these files may not display the summary properly. In this case, set <i>xml-content embed</i>=“none”.</p>
for-element-with-name/save-tokens-for-summary-processing	Sets tokenization of content in specific elements for summaries, for example, <i>dmftcontentref</i> (content of a Documentum document). Specify the maximum size of documents in bytes as the value of the attribute <i>extract-text-size-less-than</i> . Set the maximum size of index tokens for the element as the value of the attribute <i>token-size</i> .
xml-content on-embed-error	You can specify how to handle parsing errors when the <i>on-embed-error</i> attribute is set to true. Handles errors such as syntax or external entity access. Valid values: <i>embed_as_cdata</i>   <i>ignore</i>   <i>fail</i> . The option <i>embed_as_cdata</i> stores the entire XML content as a CData sub-node of the specified node. The <i>ignore</i> option does not store the XML content. For the <i>fail</i> option, content is not searchable.

Option	Description
xml-content index-as-sub-path	Boolean parameter that specifies whether the path is stored with XML content when <code>xml-content embed</code> attribute is set to true.
xml-content file-limit	Sets the maximum size of embedded XML content.
compress	Compresses the text value of specified elements to save storage space. Compressed content is about 30% of submitted XML content. Compression may slow the ingestion rate by 10-20%.
compress/for-element	Using XPath notation, specifies the XML node of the input document that contains text values to be compressed.
xml-content include-attribute-value	Specifies whether to index attribute value for xml content. Default: false.
xml-content include-processing-instruction	Specifies whether to index processing instructions in xml content. Default: false.
xml-content include-comments	Specifies whether to index comments in XML content. Default: false.

## 6.3 Configuring an index

Indexes are configured within an `indexes` element in the file `indexserverconfig.xml`. For information on modifying this file, see [“Modifying indexserverconfig.xml” on page 54](#). The path to the `indexes` element is `category-definitions/category/indexes`. Four types of indexes can be configured: fulltext-index, value-index, path-value index (sometimes called multi-path index, as it contains a list of sub-paths), and multi-path.

By default, multi-path indexes do not have all content indexed. If an element does not match one subpath, specific or general, it is not indexed. To index all element content in a multi-path index, add a subpath element on `//*`. For example, to index all metadata content, use the path `dmftmetadata/*`.

The following table displays the child elements of `node/indexes/index` that define an index.

**Table 6-2: Index definition options**

Index option	Description
path-value-index	<p>The <i>options</i> attribute of this element specifies a comma-delimited string of xDB options: GET_ALL_TEXT (indexed by its string value including descendant nodes)   SUPPORT_PHRASES (optimizes for phrase search and increases index size)   NO_LOGGING (turns off xDB transaction logging)</p> <p>The <i>path</i> attribute specifies the path to an attribute that should be indexed. The <i>path</i> attribute contains an XPath notation to a path within the input document and options for the IndexServerAnalyzer. The symbols &lt; and &gt; must be escaped.</p>
path-value-index/sub-path	<p>See “<a href="#">Subpaths</a>” on page 176. Specifies the path to an element for which the path information should be saved with the indexed value. Only applies to <i>path-value-indexes</i> that contain the IndexServerAnalyzer option INDEX_PATHS. Subpath indexes must be configured to support Documentum facets. For information on facets, see “<a href="#">About Facets</a>” on page 333.</p>
sub-path attributes	boost-value: Increases the score for hits on the subpath metadata by a factor. Default: 1.0 (no boost).
	compress: Boolean, specifies whether the content should be compressed. Only takes effect when <i>returning-contents=true</i> . Set this property to <i>true</i> when the values are enumerable. Never set it to true for datetime. Default: false.
	enumerate-repeating-elements: Boolean, specifies whether the position of the element in the path should be indexed. Used for correlated repeating attributes, for example, media objects with <i>prop_name='dimension'</i> and <i>prop_value='800x600','blue'</i> . Default: false.
	full-text-search: Specifies whether the subpath content should be tokenized and queries with ftcontains. Set to <i>true</i> if the tokens will be queried. Default: true.

Index option	Description
	include-descendants: Boolean. Default: false. If <i>true</i> , the token for this instance will have a copy of all descendant tokens, speeding up queries. Cost of this option: Lowers the indexing rate and increases disk space. Use for nodes with many small descendant nodes, such as Documentum dmftmetadata.
	leading-wildcard: Boolean, specifies whether the subpath supports leading wildcard searches, for example, owner_name ends with *utility. Cost of this option: Lowers the indexing rate and increases disk space. Default: false.
	<p>path: Path to element relative to the index path. If the path value contains a namespace, specify the xpath to it as the value of the xpath attribute. The following example has a namespace:</p> <pre data-bbox="959 910 1372 1142"> 1 path="/{http://www.w3.org/1999/ 2 02/22-rdf-syntax-ns#}RDF/{http:// www.w3.org/ 3 2004/02/skos/ core#}Concept&lt;FULL_TEXT: 4 com.emc.documentum.core.fulltext. 5 indexserver.core.index.xhive. 6 IndexServerAnalyzer: 7 GET_ALL_TEXT,SUPPORT_PHRASES, 8 INDEX_PATHS&amp;ampgt" name="Concept" </pre>
	sortable: Set to true for the results to sort on the attribute. Default: false. The attribute value-comparison must also be true. Requires reindexing.
	xpath: Value of path attribute using xpath notation without namespace. The xpath for the example in the path attribute is the following:
	<code>xpath=="/RDF/Concept"</code>
	returning-contents: Boolean, specifies whether the indexed value will be returned. For example, the user may search for documents with specific characteristics in a certain repository folder path, but the folder path does not need to be returned. Use for facets only. For information on facets, see <a href="#">"About Facets" on page 333</a> . Default: false.
	type: type of content in sub-path. Valid values: string   integer   double   date   datetime. The boolean type is not supported by xDB. Supports XQuery typed expressions such as date range. Default: string.

Index option	Description
	value-comparison: Boolean, specifies that the value in this path should be indexed. Use for comparisons such as =, >, <, starts-with. Value indexing requires additional storage, so you should not index fields that will not be searched for as comparisons or starts-with. Default: false.
	include-start-end-token-flags: Boolean, specifies whether to include start and end token flags. Supports start with, end with, and entire content query. Setting it to false can reduce index size. Default: true.
	omit-position: Boolean, specifies whether to support phrase search. Default: false. Set to <i>true</i> to reduce index size and improve ingestion speed. If omit-position=true, phrase query throws a NOT_SUPPORTED exception. To prevent exception, set xdb.lucene.strictPhraseOnly=false to translate phrase query to simple terms AND query.

### 6.3.1 Subpaths

A subpath definition specifies the path to an element. The path information is saved with the indexed value. For most Documentum applications, you do not need to modify the definitions of the subpath indexes, except for the following use cases:

- Store facet values in the index. For example:

```
<sub-path path="dmftcustom/entities/person" value-comparison="true" returning-contents="true" compress="true" />
```

- Add a subpath for non-string metadata. By default, all metadata is indexed as string type. To speed up searches for non-string attributes, add a subpath like the following. Valid types: string | integer | double | date | datetime.

```
<sub-path path="dmftmetadata//r_creation_date" type="datetime" full-text-search="false" value-comparison="true" />
```



**Note:** Starting from xPlore 1.3, boolean is no longer a supported subpath type. Any subpath with its type set to boolean in `indexserverconfig.xml` will be automatically converted to string type.

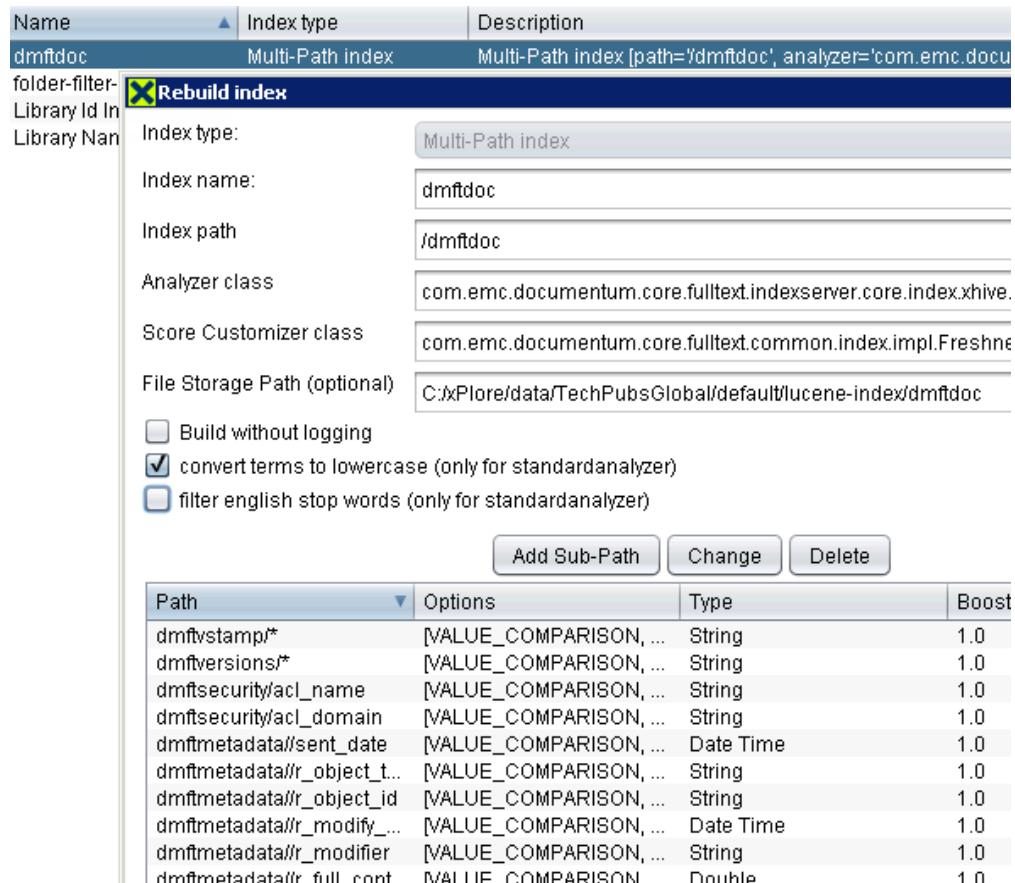
- Add paths for dmftcustom area elements (metadata or content that a TBO injects).
- Add subpaths for XML attributes. See “[XML attribute support](#)” on page 180.
- Add paths to support XQuery of XML content.

- Modify the capabilities of existing subpaths, such as supporting leading wildcard searches for certain paths, and disabling phrase search for one attribute to boost query performance. For example:

```
<sub-path path="dmftmetadata//object_name" value-comparison="true"
description="leading wildcard queries" leading-wildcard="true"/>
```

- Add subpaths for metadata that should not be indexed, for example, Documentum attributes. Set full-text-search to false.
- Add a subpath for sorting. Requires an attribute value of value-comparison="true". See "Sort support" on page 178.

 **Note:** For all subpath changes that affect existing documents, you must rebuild the index of every affected collection. To verify that your subpath was used during indexing, open the xDB admin tool and drill down to the multi-path index. Double-click **Multi-Path index** in the Index type column. You see a Rebuild Index page that lists all paths and subpaths. For example:



Name	Index type	Description
dmftdoc	Multi-Path index	Multi-Path index [path='/dmftdoc', analyzer='com.emc.docu...
dmftstamp/*	String	1.0
dmftversions/*	String	1.0
dmftsecurity/acl_name	String	1.0
dmftsecurity/acl_domain	String	1.0
dmftmetadata//sent_date	Date Time	1.0
dmftmetadata//_object_t...	String	1.0
dmftmetadata//_object_id	String	1.0
dmftmetadata//r_modify...	Date Time	1.0
dmftmetadata//r_modifier	String	1.0
dmftmetadata//r_full_c...	Double	1.0

**Figure 6-1:**

**Table 6-3: Path-value index with and without subpaths**

Feature	Without subpath	With subpath
Key set combinations	Limited	Flexible
Single key query latency	Low	High (performs better with complex predicates)
ftcontains (full-text)	Single per probe	Supports multiple ftcontains in a probe
Updates	Low overhead	High overhead
Returnable, indexed (covering) values	Yes	No

### 6.3.2 Sort support

You can configure sorting by Documentum attributes. Add a subpath in indexserverconfig.xml for each attribute that is used for sorting. Requires an attribute value of value-comparison="true".



**Note:** Reindexing is required to return documents sorted by attributes.

Follow these rules for attribute sort subpath definitions:

- Configure the most specific subpath for sorting. This insures that the index is used for lookup. For example, if the document has element root/pathA/my\_attribute and root/pathB/my\_attribute (same attribute), do not set root//some\_attribute as sortable. The following example configures an attribute for sorting using the full xpath:

```
<sub-path sortable="true" ... value-comparison="true" full-text-search="true" ...path="dmftmetadata/dm_document/r_modify_date"/>
```

- If there is more than one path to the same attribute, you must create a subpath definition for each path. Ambiguous paths may result in a slow search outside the index. In the example above, you need two subpath definitions:

```
<sub-path sortable="true" ... value-comparison="true" full-text-search="true" ...path="dmftmetadata/root/pathA/my_attribute"/>
<sub-path sortable="true" ... value-comparison="true" full-text-search="true" ...path="dmftmetadata/root/pathB/my_attribute"/>
```

- If there is only one instance of an attribute in the rendered dftxml, a shortened path notation is treated as the most specific subpath. For example, there is only one object\_name element in dftxml, so dmftmetadata//object\_name is the most specific path.

#### Specifying sort in DFC

The DFC client application can specify a set of Documentum attributes for sorting results using the API in an IDfQueryBuilder API. If the query contains an order by attribute, results are returned based on the attribute and not on the computed score.

Results are sorted by the first attribute, then sorted again by the second attribute, and so on. Two API signatures are provided:

- void addASCIIOrderByAttribute(String attrName, boolean isAsc). If isAsc is true, the results are sorted based on the specified attribute in ascending order. If false, results are sort in descending order.
- void addOrderByAttribute(String attrName, boolean isAsc, Locale locale). If the attribute is a string type, and the locale is not specified, the locale of the query is used. The locale parameter is ignored in DQL queries.

## Sort debugging

You can use the Query debug and Optimizer tools in xPlore administrator to troubleshoot sorting. When an index is properly configured and used for the query, you see the following in the Query debug pane:

```
Found an index to support all order specs. No sort required.
```

If you do not see this confirmation, check the Optimizer pane. Find the following section:

```
<endingimplicitmatch indexname="dmftdoc">
```

In the following example, there are two order-by clauses in the query. The first fails because there is no sub-path definition for sorting.

```
<endingimplicitmatch indexname="dmftdoc">
<LucenePlugin>
<ImplicitIndexOptimizer numconditionsall="1"
    numorderbyclausesall="2">
    <condition accepted="true" expr="child::dmftmetadata/
        descendant-or-self::node()/child::object_name
        [. contains text rule]"/>
    <orderbyclause accepted="true" expr="child::dmftmetadata/
        descendant-or-self::node()/child::object_name"/>
    <orderbyclause accepted="false" reason="
        No exact matching subpath configuration found that
        matches order-by clause" expr="child::dmftmetadata/
        descendant-or-self::node()/child::r_modify_data"/>
</ImplicitIndexOptimizer>
<numacceptedandrejected numconditionsaccepted="1"
    numconditionsrejected="0" numorderbyclausesaccepted="1"
    numorderbyclausesrejected="1"/>
</LucenePlugin>
<conditions numaccepted="1">
    <pnodecondition>
        child::dmftmetadata/descendant-or-self::
            node()/child::object_name[. contains text rule]
    </pnodecondition>
    <externallyoptimizedcondition accepted="true">
        child::dmftmetadata/descendant-or-self::node()/
            child::object_name[. contains text rule]
    </externallyoptimizedcondition>
</conditions>
<orderbyclauses numaccepted="1">
    <orderbyclause accepted="true">child::dmftmetadata/
        descendant-or-self::node()/child::object_name
    </orderbyclause>
    <orderbyclause accepted="false">child::dmftmetadata/
        descendant-or-self::node()/child::r_modify_data
    </orderbyclause>
```

```
</orderbyclauses>
</endingimplicitmatch>
```

The information displays that the order by clause is not accepted, which means it would not take effect. The probable reason is a typo in the sub-path definition in indexserverconfig.xml: The proper sub-path is dmftmetadata//r\_modify\_date.

### 6.3.3 XML attribute support

By default, xPlore only indexes XML elements but not XML attributes. To configure xPlore to index XML attributes so that attribute values can be searched, you must explicitly add them to the subpath definitions in indexserverconfig.xml.

In the following example, the XML attribute *dmfttype* is added as a subpath definition for indexing.

```
<sub-path path="dmftmetadata//r_object_id/@dmfttype"
value-comparison="true"/>
```



**Note:** Attributes of descendant elements are not included, even if *include-descendants* is true. You must also explicitly add subpath definitions for descendant attributes for them to be indexed.

Once the index is rebuilt for the changes to take effect, you can search for attribute values using XQuery expressions; for example:

```
for $i score $s in /dmftdoc[dmftmetadata//r_object_id/@dmfttype
ftcontains 'dmid' with stemming] order by $s descending return
<d> {$i/dmftmetadata//r_object_id} { $i/dmftmetadata//object_name }
{ $i/dmftmetadata//r_modifier } </d>
```

Indexing and querying XML attribute values in the XML content has the following considerations:

- When you turn on XML zone search (index-as-sub-path = true), XML structure is preserved along with the content of the elements that is embedded in dmftxml, therefore, XML attributes in the XML content are indexed in the same way as those in the metadata, and no additional steps are needed.
- When you turn off XML zone search (index-as-sub-path = false), you need to set *include-attribute-value* to *true* on the *xml-content* element so that xPlore will extract attribute values from the XML structure for indexing.

## 6.4 Creating custom indexes

xPlore indexes full-text content and metadata for all objects in the repository, so you do not need to set up indexes. Create a custom index for the following:

- A subpath for each facet used by the client application
- Custom, non-repository metadata or content added to your documents during indexing. In Documentum environments, use a TBO.

Set up indexes before content is ingested. If you add an index on existing content, the content must be reingested.

The Documentum index agent generates metadata in a data input format called dftxml. Each indexable object type is represented in dftxml, along with its indexable attributes, in a predictable path within the dftxml representation. Refer to ["Extensible Documentum DTD" on page 413](#) for additional information on dftxml.

Indexes are defined within indexserverconfig.xml, in *xplore\_home/config*, within a category definition. Shut down all xPlore instances before changing this file. To enhance performance and reduce storage, you can specify categories and attributes that are not indexed.

### Indexing new attributes

You can add path-value indexes to indexserverconfig.xml for new data types so that documents of the new type can be ingested without rebuilding the index. Shut down all xPlore instances before changing this file. If you add a new attribute for documents that have already been indexed, rebuild the index.

## 6.5 Managing indexing in xPlore administrator

You can perform the following administrative tasks in xPlore administrator.

- *View indexing statistics:* Expand an instance in the tree and choose **Indexing Service**. Statistics are displayed in the right panel: tasks completed, with a breakdown by document properties, and performance.
- *Configure indexing across all instances:* Expand **Services > Indexing Service** in the tree. Click **Configuration**. You can configure the various options described in ["Document processing and indexing service configuration parameters" on page 402](#). The default values have been optimized for most environments.
- *Start or stop indexing:* Select an instance in the tree and choose **Indexing Service**. Click **Enable** or **Disable**.
- *View the indexing queue:* Expand an instance in the tree and choose **Indexing Service**. Click **Operations** to display the queue. You can cancel any indexing batch requests in the queue.



**Note:** This queue is not the same as the index agent queue. You can view the index agent queue in the index agent UI or in Documentum administrator.

## 6.6 Troubleshooting indexing

You can use reports to troubleshoot indexing and content processing issues. See “[Document processing \(CPS\) reports](#)” on page 350 and “[Indexing reports](#)” on page 350 for more information on these reports.

### Testing upload and indexing

If xPlore administrator is running on the same instance as an index service, you can test upload a document for indexing.



**Note:** If you are testing Documentum indexing before migration, run the ACL replication script to update the security index. “[Manually updating security](#)” on page 64.

To do test upload in xPlore administrator, expand the **Diagnostic and Utilities** tree and select **Upload Testing Document**. You can test upload in the following ways:

- **Local File:** Navigate to a local file and upload it.
- **Remote File:** Enter the path to a remote file that is accessible from the xPlore administrator host.
- **Specify raw XML:** Click **Specify raw XML**, and type raw XML such as dftxml for testing.

For the **Content Type** option (local or remote file), specify a MIME type such as application/msword, application/pdf, text/plain, or text/xml.

When you click the object name, you see the dftxml. This dftxml rendition is template-based, not generated by the Documentum index agent. There can be slight differences in the dftxml when you submit the document through the Documentum index agent. To verify remote CPS, you start with a sample dftxml file, then edit the *dmcontentref* element. Set it as a path to the shared storage, then paste the dftxml in the text box.

Results are displayed. If the document has been successfully uploaded, it is available immediately for search unless the indexing load is high. The object name in xPlore is created by concatenating the file name and timestamp.

### Checking network bandwidth and latency

Bandwidth or latency bottlenecks can degrade performance during the transfer of content from a source file system to xPlore. CPS reports measure indexing latency. Validate that file transfers take place with the expected speed. This issue is seen more frequently in virtual environments.

## Checking the indexing log

The xPlore log `dsearch.log` is located in the `logs` subdirectory of the WildFly deployment directory.

## Troubleshooting high save-to-search latency

The following issues can cause high latency between the time a document is created or updated and the time the document is available for search:

- Index agent is down

Detect this problem by monitoring the size of the index agent queue. Use xPlore administrator to determine whether documents were sent for ingestion. For example, the Documents ingested per hour report shows 0 for DocCount when the index agent is down.

*Workaround:* Configure multiple index agents for redundancy. Monitor the index agents and restart when they fail.

- CPS restarts frequently

Under certain conditions, CPS fails while processing a document. xPlore restarts the CPS process, but the restart causes a delay. Restart is logged in `cps.log` and `cps_daemon.log`. For information on these logs, see “[CPS logging](#)” on page 359.

- Large documents tie up ingestion

A large document in the ingestion pipeline can delay smaller documents that are further back in the queue. Detect this issue using the Documents ingested per hour report in xPlore administrator. (Only document size averages are reported.)

If a document is larger than the configured maximum limit for document size or text size, the document is not indexed. The document metadata are indexed but the content is not. These documents are reported in the xPlore administrator report *Content too large to index summary report by domain*.

*Workarounds:* Attempt to refeed a document that was too large. Increase the maximum size for document processing or set `cut_off_text` to true in PrimaryDsearch\_local\_configuration.xml. This file is located in `xplore_home/dsearch/cps/cps_daemon`. See “[Maximum document and text size](#)” on page 124.

- Ingestion batches are large

During periods of high ingestion load, documents can take a long time to process. Review the ingestion reports in xPlore administrator to find bytes processed and latency. Use `dsearch.log` to determine when a specific document was ingested.

*Workaround:* Set up a dedicated index agent for the batch workload.

- Insufficient hardware resources

If CPU, disk I/O, or memory are highly utilized, increase the capacity. Performance on a virtual server is slower than on a dedicated host. For a comparison or performance on various storage types, see “[Disk space and storage](#)” on page 369.

## Changes to index configuration do not take effect

If you have edited indexserverconfig.xml, your changes are not applied unless the system is restarted. Some changes, such as adding or modifying a subpath do not take effect on a restart but only when you rebuild the indexes of the collection.

[“Modifying indexserverconfig.xml” on page 54](#) describes the procedure to view and update the index configuration.

If you have changed configuration using xPlore administrator, and your system crashed before the changes were flushed to disk, repeat your configuration changes and restart xPlore.

See also: [“Running the standalone data consistency checker” on page 185](#).

## Resolving attribute datatype conflict

To speed up searches for non-string attributes, you can define a subpath specifying the attribute type: integer, double, date, or datetime. However, a conflict occurs in the following conditions:

- You changed the attribute type in the subpath definition to an incompatible type. For example, you can change an attribute from string type to any other type, but you cannot change an attribute from double to date type.

*And*
- Some objects have already been indexed.

When new objects are indexed, the system throws the exception: XhiveException: INDEX\_VALUE\_FORMAT\_ERROR. The exception appears in dsearch.log, in indexagent.log, and in Documentum Administrator, in the Index queue page.

For example, in dsearch.log:

```
2012-09-25 03:42:16,516 ERROR [IndexWorkerThread-2]
c.e.d.c.f.i.core.index.plugin.XhivePlugin
- Failed to ingest documents: 0900107f8000425c; Exception:
com.xhive.error.XhiveException: INDEX_VALUE_FORMAT_ERROR:
Illegal value "2013-01-01T09:00:00" for type DOUBLE in index dmftdoc
```

After changing sub-path configuration, you need to rebuild the index if there is data in the collection. Therefore, if the existing data has conflicts with the new sub-path configuration, rebuilding fails.

To resolve the conflict, align the sub-path configuration correctly with the attribute type. After aligning, rebuild the index, then start indexagent indexing normally.

## 6.7 Running the standalone data consistency checker

The standalone data consistency checker has two functions:

- Checks data consistency for all collections in a specific domain.
- Detects tracking database corruption and rebuilds it, then reconciles it with the corresponding collection.

For example, for the index agent log error *Duplicate document ID*, run the consistency checker to see whether the tracking DB is out of sync with the index. Make sure there is no indexing activity when running this tool.



### Notes

- Do not confuse data consistency check with database consistency check. See “[Domain and collection menu actions](#)” on page 191.
- Do not check consistency during migration or index rebuild. Run only one instance of the tool at a time.

The consistency checker is invoked from the command line. The tool checks consistency with the following steps. The batch size for a list is configurable (default: 1000). Batches are processed until all documents have been checked.

1. For each collection in the domain, the tool queries the tracking database to find the number of document IDs.
2. Document IDs are returned as a list from the tracking DB.
3. Document IDs are returned as a list from the corresponding collection.
4. The list sizes are compared. Differences are reported.
5. Set `is_fix_trackDB` to true, it inserts the missing entries in the tracking database. It deletes the entries in the database that do not exist in the collection. It also validates the library-path in the database entry.

Detailed steps are described as follows:

1. Back up your domain or collection.
2. Stop all indexing activity on the instance. Set the target collection, domain, or federation to read-only or maintenance mode.
3. Invoke the checker using CLI. (See “[Using the CLI](#)” on page 227.) The default batch size is 1000. Edit the script to change the batch size. Syntax (on one line):

```
xplore checkDataConsistency unit, domain, collection,  
is_fix_trackDB, batch-size, report-directory
```

Valid values:

- `unit`: *collection* or *domain*

- *domain*: domain name.
- *collection*: Collection name (null for domain consistency check)
- *is\_fix\_trackDB*: *true* or *false*. Set to *false* first and check the report. If indexing has not been turned off, inconsistencies are reported.
- *batch-size*: Numeric value greater than or equal to 1000. Non-numeric, negative, or null values default to 1000.
- *report-directory*: Path for consistency report. Report is created in a subdirectory *report-directory/time-stamp/domain\_name|collection\_name*. Default base directory is the current working directory.

Windows example: Checks consistency of a *default* collection in *defaultDomain* and fixes the trackingDB:

```
xplore "checkDataConsistency 'collection', 'defaultDomain','default1' true,  
'2000', 'C:\\tmp' "
```

Checks consistency of all collections in *defaultDomain* and fixes the trackingDB:

```
xplore "checkDataConsistency 'domain', 'defaultDomain',null, true,  
'2000', 'C:\\tmp' "
```

Linux example: Checks consistency of *defaultDomain*:

```
./xplore.sh checkDataConsistency " 'domain', 'defaultDomain', 'null', 'true',  
'1000', 'export/space1/temp' "
```

#### 4. View the report in the current working directory.

##### *Check database consistency online*

Consistency check usually takes a long time to complete and requires you stop any other activities such as indexing and query services. To reduce system downtime, use the following steps to check on backup files:

1. Create a consistency check environment. Perform a new xPlore installation on a different host other than the xPlore production environment. You do not need to deploy any Dsearch, CPS, or index agent instances on that host.
2. In the xPlore production environment, perform a full federation backup of the xPlore deployment (single or multiple instances) and copy the backup folder to the consistency check environment.
3. In the consistency check environment, perform consistency check as needed.
  - a. In *xplore\_home/dsearch/xhive/admin*, run *XHCommand.bat* (Windows) or *XHCommand* (Linux).
  - b. Run *check-\** command with *-<backup-workdir> -<backup parameters>*; for example:
    - To check all options on database:

```
check-database -d xhivedb --backup-workdir  
C:\\xPlore\\dsearch\\backup\\federation\\2013-09-22-22-27-06  
--backup C:\\xPlore\\dsearch\\backup\\federation\\2013-09-22-22-27-06
```

```
\FederationBackup.dmp --check-domnode --check-pagesowner
--check-indexes --check-adminpages --check-segments
```

- To check indexes on given collection:

```
check-librarychild -d xhivedb --backup-workdir
C:\xPlore\dsearch\backup\federation\2013-09-22-22-27-06
--backup C:\xPlore\dsearch\backup\federation\2013-09-22-22-27-06
\FederationBackup.dmp --check-indexes /DOCREPO/dsearch/Data/default
```

## 6.8 Indexing APIs

Access to indexing APIs is through the interface `com.emc.documentum.core.fulltext.client.IDSearchClient`. Each API is described in the javadocs. The following topics describe the use of indexing APIs.

### 6.8.1 Route a document to a collection

You can route a document to a collection in the following ways:

- A custom routing class. See “[Creating a custom routing class](#)” on page 187.
- Index agent configuration. See “[Mapping Server storage areas to collections](#)” on page 90 and “[Sharing content storage](#)” on page 88.

#### 6.8.1.1 Creating a custom routing class

xPlore defines an interface for routing a document to a collection: `IFtIndexCollectionRouting` in the package `com.emc.documentum.core.fulltext.client.index.custom`. You can provide a class that implements the interface and specify this class name in the xPlore configuration file `indexserverconfig.xml`. Then xPlore invokes the custom class for routing.

You can use custom routing for all domains or for specific domains. The collection determined by the routing class takes precedence over a collection that is specified in the indexing API.

1. Stop the xPlore instances.
2. Register your custom routing class in `indexserverconfig.xml`, which is located in the xPlore config directory. Add the following element between the `system-metrics-service` and `admin-config` elements. (The xPlore server must be stopped before you add this element.)

```
<customization-config>
  <collection-routing class-name="SimpleCollectionRouting">
  </collection-routing>
</customization-config>
```

3. Create your custom class. (See example.) Import `IFTIndexRequest` in the package `com.emc.documentum.core.fulltext.common.index`. This class encapsulates all aspects of an indexing request:

```
public interface IFTIndexRequest
{
    String getDocId();
```

```

        long getRequestID ();
        FtOperation getOperation (); //returns add, update or delete
        String getDomain ();
        String getCategory ();
        String getCollection ();
        IFtDocument getDocument (); //returns doc to be indexed
        public String getClientId();
        void setCollection(String value);
        public void setClientId(String id);
    }
}

```

### 6.8.1.2 SimpleCollectionRouting example

This example routes a document to a specific collection based on Documentum version.

The sample Java class file in the SDK/samples directory assumes that the Documentum index agent establishes a connection to the xPlore server. Place the compiled class SimpleCollectionRouting in the Documentum index agent classpath, for example, *xplore\_home/<wildfly\_version>/server/DctmServer\_Indexagent/deployments/IndexAgent.war/WEB-INF/classes*.

This class parses the input DFTXML representation from the index agent. The class gets a metadata value and tests it, then routes the document to a custom collection if it meets the criterion (new document).

#### Imports

```

import com.emc.documentum.core.fulltext.client.index.*;
custom.IFtIndexCollectionRouting;
import com.emc.documentum.core.fulltext.client.index.FtFeederException;
import com.emc.documentum.core.fulltext.client.common.IDSearchServerInfo;
import com.emc.documentum.core.fulltext.common.index.IFtIndexRequest;
import com.emc.documentum.core.fulltext.common.index.IFtDocument;

import java.util.List;
import javax.xml.xpath.*;
import org.w3c.dom.*;

```

#### Required method setEssServerInfo

The index agent environment sets the xPlore server info. You can implement this method simply without creating the connection:

```

public void setEssServerInfo(IDSearchServerInfo info)
{
    m_serverInfo = info;
}

IDSearchServerInfo m_serverInfo;

```

#### Add variables for routing

```

private boolean m_updated = false;
private static final String s_collection = "superhot";
private String m_version = null;

```

## Parse the metadata from dftxml

This method parses the dftxml representation of the document and metadata, which is passed in from the Documentum index agent. The Documentum version is returned to setCustomCollection().

```
private String parseDocumentVersion(Document inputdoc)
throws XPathExpressionException
{
    String version = null;
    XPathFactory factory = XPathFactory.newInstance();
    XPath xpath = factory.newXPath();
    XPathExpression expr = xpath.compile("//r_version_label/text()");
    Object result = expr.evaluate(inputdoc, XPathConstants.NODE);
    Node versionnode = (Node) result;
    version = versionnode.getNodeValue();
    System.out.println("version: " + version);
    m_version = version;
    return m_version;
}
```

## Add routing logic

This method calls parseDocumentVersion() to get the version for routing. It sets the custom collection if the metadata meets the criterion (new collection).

```
private boolean setCustomCollection(IFtIndexRequest request)
{
    assert request != null;
    IFtDocument doc = request.getDocument();
    Document xmldoc = doc.getContentToIndex();
    try
    {
        String version = parseDocumentVersion(xmldoc);
        if (version.equals "1.0")
        {
            request.setCollection(s_collection);
            m_updated = true;
        }
        } catch (XPathExpressionException e)
        {
            new FtFeederException(e);
        }
        return m_updated;
    }
```

## Set the custom collection

This method determines whether the document is being added (not updated or deleted). The method then calls setCustomCollection to get the version and route the document to the appropriate collection.

```
// Return true after the collection name has been altered for any request
// Otherwise returns false.

public boolean updateCollection(List<IFtIndexRequest> requests) throws
FtFeederException
{
    assert m_serverInfo != null;
    assert requests != null;

    for ( IFtIndexRequest request : requests )
    {
        if (request.getOperation().toString().equals("add"))
    }
```

```
    setCustomCollection(request);
}
}
return m_updated;
}
```

## Chapter 7

# Index Data: Domains, Categories, and Collections

## 7.1 Domain and collection menu actions

### Execute XQuery

You can query a domain or collection with **Execute XQuery** in xPlore administrator. Enter your XQuery expression in the input area. The options **get query plan** and **get optimizer debug** are used to provide information to OpenText Global Technical Services.

### Check database consistency online

Database consistency check determines whether there are any corrupted or missing files such as configuration files or Lucene indexes. Lucene indexes are checked to see whether they are consistent with the xDB records: tree segments, xDB page owners, and xDB DOM nodes.



**Note:** Do not confuse database consistency check with data consistency check. See “[Running the standalone data consistency checker](#)” on page 185.

Perform the database consistency check before backup and after restore.



**Note:** You must set the domain to maintenance mode before running this check.

Select the following options to check. Some options require extensive processing time:

- Segments and admin structures: Efficient check, does not depend on index size.
- Free and used pages consistency: Touches all pages in database.



**Note:** When the result from this check is inconsistent, run it two more times.

- Pages owner: Touches all pages in database.
- Indexes: Traverses all the indexes and checks DOM nodes referenced from the index.
- Basic checks of indexes: Efficient check of the basic structure. The check verifies that the necessary Lucene index files exist and that the internal xDB configuration is consistent with files on the file system.
- DOM nodes: Expensive operation, accesses all the nodes in the database.

The basic check of indexes inspects only the external Lucene indexes. The check verifies that the necessary files exist and that the internal xDB configuration is consistent with files on the file system. This check runs much faster than the full consistency check.

## 7.2 Managing domains

A domain is a separate, independent, logical, or structural grouping of collections. Domains are managed through the Data Management screen in xPlore administrator. The Documentum index agent creates a domain for the repository to which it connects. This domain receives indexing requests from the repository.

### New domain

Select Data Management in the left panel and then click **New Domain** in the right panel. Choose a default document category. (Categories are specified in `indexserverconfig.xml`.) Choose a storage location from the dropdown list. To create a storage location, see “[Creating a storage location](#)” on page 204.

A Documentum index agent creates a domain for a repository and creates ACL and group collections in that domain.

### New Collection

Create a collection and configure it. See “[Changing collection properties](#)” on page 198.

### Configuration

The document category and storage location are displayed (read-only). You can set the runtime mode as normal (default) or maintenance (for a corrupt domain). The mode does not persist across xPlore sessions; mode reverts to runtime on xPlore restart.

### Back up, detach or attach a domain

For information on backing up a domain, see “[Performing a native xDB backup in xPlore Administrator](#)” on page 217.



**Note:** Do not use detach to move a domain index. Instead, change the binding for each collection. To avoid corruption, do not detach or attach a domain during index rebuild.

### Delete domain

If you are unable to start xPlore, with a log entry that the domain is corrupted, force recovery. Add a property `force-restart-xdb=true` in `xplore_home/<wildfly_version>/server/<instance_name>/deployments/dsearch.war/WEB-INF/classes/indexserver-bootstrap.properties`.

Consider removing the domain if one or both of the following apply:

- The domain is no longer needed.
- The domain library is corrupted. This is quite rare, since no information is stored in the domain library. If a collection is corrupted, consider deleting only that collection.

If you want to delete a domain with detached collections, off-line collections, or search-only collections, and these collections are no longer needed, then delete them before deleting the domain.

To back up the domain before deletion, you need to back up the entire config folder, data folder, and wal folder.

## 7.3 Delete a corrupted domain

If one or more segments of a domain are corrupted, only after the user encounters a DATA\_CORRUPTION exception, the state of the domain in xPlore Admin Console will be set to off\_line, which is a runtime state and does not persist.

Consider deleting a domain only when it is not needed or the domain segment is corrupted. If only a collection is corrupted, delete that collection directly.

To delete a corrupted domain, do the following:

1. Force recovery if the domain is corrupted. Add a property force-restart-xdb=true in the file indexserver-bootstrap.properties in <xplore\_home>/<wildfly\_version>/server/<instance\_name>/deployments/dsearch.war/WEB-INF/classes.
2. Log in to xPlore administrator and open the domain in **Data Management**. Click **Detach** to force detach the domain.
3. Stop all xPlore instances.
4. Back up the <xplore\_home>/config folder, the <xplore\_home>/data/<domain\_name> folder, and the wal folder.
5. Remove the domain element from indexserverconfig.xml, like the following:

```
<domain ...name="my_domain">
  ...
</domain>
```

6. Delete the segment elements for the domain in XhiveDatabase.bootstrap in xplore\_home/config. Search on the domain name. For example, delete the segment element with library-path that contains GlobalOps library-path or id attribute:

```
<segment library-id="0" library-path="/Global0ps/dsearch/
ApplicationInfo/acl" ...
id="Global0ps#dsearch#ApplicationInfo#acl">
<file id="10" path="C:\xPlore\data\Global0ps\acl\xhivedb-
Global0ps#dsearch
```

```
#ApplicationInfo#ac1-0.XhiveDatabase.DB"/>
<binding-server name="primary"/>
</segment>
```

7. Delete the domain folder under xplore\_home/data, for example, xplore\_home/data/GlobalOps.
8. Restart the xPlore primary and then secondary instances.

## 7.4 Configuring categories

A category defines a class of documents and their XML structure. The category definition specifies the processing and semantics that are applied to the ingested XML document. You can specify the XML elements that have text extraction, tokenization, and storage of tokens. You also specify the indexes that are defined on the category and the XML elements that are not indexed. More than one collection can map to a category. xPlore manages categories.

The default categories include dftxml (Documentum content), security (ACL and group), tracking database, metrics database, audit database, and thesaurus database.

When you create a collection, choose a category from the categories defined in indexserverconfig.xml. When you view the configuration of a collection, you see the assigned category. It cannot be changed in xPlore administrator. To change the category, edit indexserverconfig.xml.

You can configure the indexes, text extraction settings, and compression setting for each category. The paths in the configuration file are in XPath syntax and refer to the path within the XML representation of the document. (All documents are submitted for ingestion in an XML representation.) Specify an XPath value to the element whose content requires text extraction for indexing.

**Table 7-1: Category configuration options**

Option	Description
category-definitions	Contains one or more category elements.
category	Contains elements that govern category indexing.
properties/property track-location	Specifies whether to track the location (index name) of the content in this category. For Documentum dftxml representations of documents, the location is tracked in the tracking DB. Documentum ACLs and groups are not tracked because their index location is known.

### Avoiding duplicated index requests

The object\_version\_path property in indexserverconfig.xml enables you avoid duplicated index requests by defining an XPath expression (integer value) that

specifies the object version. With this property defined, xPlore only handles requests with versions equal to or higher than specified object version.

```
<category name="dftxml">
  <properties>
    ...
    <property name="track-location" value="true" />
    <property name="enable-freshness-score" value="true" />
    <property name="object_version_path" value="xpath_expression" />
    ...
  </properties>
  ...
</category>
```

If clients submit update and delete requests concurrently, xPlore only handles the delete requests, marking update request completed directly.

## 7.5 Managing collections

### 7.5.1 About collections

A collection is a logical group of XML documents that is physically stored in an xDB detachable library. A collection represents the most granular data management unit within xPlore. All documents submitted for indexing are assigned to a collection. A collection generally contains one category of documents. In a basic deployment, all documents in a domain are assigned to a single default collection. You can create subcollections under each collection and route documents to user-defined collections.

A collection is bound to a specific instance in read-write state (index and search, index only, or update and search). A collection can be bound to multiple instances in read-only state (search-only).

#### Viewing collections

To view the collections for a domain, choose **Data Management** and then choose the domain the left pane. In the right pane, you see each collection name, category, usage, state, and instances that the collection is bound to.

There is a red X next to the collection to delete it. For xPlore system collections, the X is grayed out, and the collection cannot be deleted.

#### Viewing collection properties

Choose **Data Management** and drill down to the collection in the left pane. In the content pane, you see the following information about the collection:

- Library path in xDB.
- Document category. For more information, see “[Documentum domains and categories](#)” on page 27.
- xPlore instances the collection is bound to.

- State: Valid states:*index and search*, *index only*, *update\_and\_search*, *search only*, and *off\_line*.



**Note:** You cannot back up a collection in the offline state. You must detach it or bring it online.

- Usage: Type of xDB library. Valid types: data (index), ApplicationInfo and SystemInfo.
- Current size on disk

## Viewing collection contents

The total number displayed in Collection Contents pane is the number of documents in the collection and its subcollections. For subcollections, the number of documents in the subcollection is displayed.

## Viewing a document in a collection

Click **Name** for an individual document to view the XML representation (dftxml) of the document.

### 7.5.2 Planning collections for scalability

A collection is a logical grouping of tokenized content and associated full-text indexes within a domain. For example, you have a collection that indexes all email documents. For Documentum environments, the Documentum index agent creates a domain for each source repository, and documents are indexed to collections within that domain.

A collection contains documents of a single category. A category can be mapped to multiple collections. A document category definition in `indexserverconfig.xml` describes what is indexed within the documents that are submitted

Specify the target collection for documents using one of the following methods. They have the following order of precedence in xPlore, with highest first:

- Custom routing class. See “[Creating a custom routing class](#)” on page 187.
- API indexing option or partition mapping in the Documentum index agent. See “[Sharing content storage](#)” on page 88.
- Backend collection routing, if you have enabled it. See “[Route a document to a collection](#)” on page 187.
- Default collection that is created for each domain.

When the target collection is not specified, documents are indexed to the collections in round-robin order. The documents are passed to the instance with the specified collection. Only collections with the state *index* or *index and search* can receive documents.

You can create additional collections for high ingestion rates and then move those collections to a parent collection. A single top-level collection hierarchy provides better search performance than multiple top-level collections.

### 7.5.3 Uses of subcollections

Create subcollections for the following uses:

- Create multiple top-level collections for migration to boost the ingestion rate. After ingestion completes, move the temporary collection to a parent collection. The temporary collection is now a subcollection. The parent and subcollections are searched faster than a search of multiple top-level collections.
- Create subcollections for data management. For example, you create a collection for 2011 data with a subcollection to store November data.

The following restrictions are enforced for subcollections, including a collection that is moved to become a subcollection:

- Subcollections cannot be detached or reattached. For example, a path-value index is defined with no subpaths, such as the folder-list-index.
- Subcollections cannot be backed up or restored separately from the parent.
- Subcollections must be bound to the same instance as the parent.
- Subcollection state cannot contradict the state of the parent. For example, if the parent is search\_only, the subcollection cannot be index\_only or index\_and\_search. If the parent is indexable, the adopted collection cannot be search-only.

Exception: When the parent collection is in update\_and\_search or index\_and\_search state, subcollections can be any state.



**Note:** By default, no fulltext index is created on sub-collections. All indexes are added to the parent collection for the index in xPlore. When querying a sub-collection directly, ensure the appropriate index has been created.

### 7.5.4 Adding or deleting a collection

1. Choose a domain and then choose **New collection**.
2. Set properties for the new collection:
  - Collection name
  - Parent domain
  - Usage: Type of xDB library. Valid types: data (index) or ApplicationInfo. New collections cannot be added for acl, group, thesaurusdb, and metricsdb.
  - Document category: Categories are defined in indexserverconfig.xml.
  - Binding instance: Existing instances are listed.

To change the binding of a collection, see “[Changing collection properties](#)” on page 198.

- Storage location: Choose a storage location from the dropdown list. To define a storage location, see “[Creating a storage location](#)” on page 204.
- 3. To create a subcollection, click the parent collection in the navigation pane and then click **New Subcollection**. The storage location is the same as the location for the parent.
- 4. To delete a collection, choose a domain and then click **X** next to the collection you wish to delete.



**Note:** When a collection is deleted, the corresponding segment files are deleted and cannot be recovered.

### 7.5.5 Changing collection properties

1. Select a collection and then choose **Configuration**.  
The **Edit collection** screen displays information such as state, binding instance, and final merge scheduler.
2. Configure state: index and search, index only, update and search, or search only.
  - *index and search* is the default state when a new collection is created. A collection can have only one binding that is *index\_and\_search*.
  - Use *index only* to repair the index. You cannot query a collection that is set to *index only*.
  - Use *update and search* for read-only collections that have updates to existing content or metadata. You cannot add new content to the collection.
  - Use *search only* (read-only) on multiple instances for query load balancing and scalability.
3. Change binding to another xPlore instance:
  - a. Set the state of the collection. If the collection state is *index\_and\_search*, *update\_and\_search*, or *index\_only*, you can bind to only one instance. If the collection state is *search\_only*, you can bind the collection to multiple instances for better resource allocation.
  - b. Choose a **Binding instance**.

Limitations:

- If a binding instance is unreachable, you cannot edit the binding.
- You cannot change the binding of a subcollection to a different instance from the parent collection.
- To change the binding on a failed instance, restore the collection to the same instance or to a spare instance.

## 7.5.6 Routing documents to a specific collection

There are five ways to route documents to a specific collection. Routing is applied in the following order of precedence, with custom routing applied in highest priority.

- Custom routing class. Use for complicated routing logic before migration. See “[Creating a custom routing class](#)” on page 187.
- Documentum index agent: Map a file store to a collection before migration. See “[Sharing content storage](#)” on page 88.
- Collection hints specified in `indexagent.xml`.
- Predefined routing classes: Use to route documents in the following order, but you can also use other classes to route based on document attributes:
  1. Check default collection to confirm it is valid.
  2. Check collection hash to see whether one can be picked up.
  3. Route in round-robin order among all collections.

## 7.5.7 Routing to specific collections using routing classes

You can use xPlore predefined routing classes to route documents based on document type, document size or other parameters. The routing classes are derived from a backend-collection-routing configuration, which uses the `FtAbstractCollectionRouting` class. Backend collection routing classes are implemented separately from the IndexAgent routing mechanism.



**Note:** If IndexAgent routing mechanisms are enabled, the backend routing classes are disabled.

Backend collection routing is dependent on the following, in order of priority:

1. Collection hint from xPlore client side, such as IndexAgent set collection hint
2. Collection record in trackingDB
3. Collection calculated from the routing class

To route objects using predefined routing classes, you must configure the property of the object, such as field value range for document size, and the rule and condition to dispatch the object.

Routings with predefined routing classes are described in “[Routing on XPath](#)” on page 199 and related routing topics, with a sample provided for XPath routing and property configurations provided for other routings. In addition, implementation of a customized routing class is described in “[Customizing a routing class](#)” on page 201.

### Routing on XPath

To route objects with a specific XPath expression, use the `FtCollectionRoutingOnXPath` class. You can construct the XPath to configure complex

routing criteria, such as routing on multiple field values. The XPath expression should return a boolean value.

In the following example, if the object's `i_folder_id` equals 3456789, it will be routed to collection 1, and if it equals 456789, it will be routed to collection 2.

```
<backend-collection-routing class-name="FtCollectionRoutingOnXPath"
default-collection="default" category ="dftxml">
<properties>
</properties>
<collection-routing>
<rule condition="boolean(/dmftdoc//i_folder_id='3456789' )"
collection="collection1"/>
<rule condition="boolean(/dmftdoc//i_folder_id='456789' )"
collection="collection2"/>
<rule condition="boolean(/dmftdoc//i_folder_id='4567890' and
/dmftdoc//object_name='test')" collection="collection3"/>
</collection-routing>
</backend-collection-routing>
```

## Routing on field value

To route objects based on the value of one object field, use the `FtCollectionRoutingOnFieldValues` class. Configure the field value for collection mapping using rules and conditions from the XPath routing example:

```
<properties>
<property name="routing-field-xpath" value="dmftmetadata//file_store"/>
</properties>
```

You can configure collection routing rules as shown in the following example:

```
<collection-routing>
<rule condition="file_store_01" collection="collection1"/>
<rule condition="file_store_02" collection="collection2"/>
</collection-routing>
```

Routing is based on the value of the incoming object's `dmftmetadata//file_store` field. If the object's `file_store` equals `file_store_01`, xPlore routes the object to collection 1. If an object's `file_store` equals `file_store_02`, xPlore will route it to collection 2. Otherwise, xPlore will route objects to the configured default collection.

For routing on field value, and for all other routings, you can configure multiple keys in one condition, separated by a value-splitter (default value is comma):

```
<properties>
<property name="value-splitter" value="," />
</properties>
```

## Routing on field value range

To route objects based on continuous value, or content size, use the `FtCollectionRoutingOnFieldValueRange` class as shown in the following example:

```
<properties>
<property name="routing-field-xpath" value=
"dmftmetadata//r_content_size"/>
<property name="value-type" value="Integer" />
<property name="value-range-separator" value="~" />
</properties>
```

You can configure collection routing rules for a value range as shown in this example:

```
<collection-routing>
  <rule condition="40000~80000" collection="collection1"/>
  <rule condition="80000~100000" collection="collection2"/>
</collection-routing>
```

xPlore uses string comparison for specified values and values in the dftxml file. For example, "abc" is smaller than "bbc", "123" is smaller than "223", but "12" is smaller than "9". Zero-padded values are not supported in the indexserverconfig.xml or dftxml files.

To specify non-string type, you need to specify the value-type in the property section. The following values are supported: Integer, Double, Datetime and String. Default: String. For Datetime, unify the UTC time (yyyy-MM-dd'T'HH:mm:ss). If unacceptable data is entered in the dftxml file, string comparison is used to determine the collection.

The configured mapper rule is inclusive. If one object can be routed by two rules, then the first rule is used. If invalid collections are configured, that rule is skipped during routing.

## Routing on field value limited with regular expression

For special field value requirements, you can route objects using the *FtCollectionRoutingOnFieldValueWithRegularExpression* class. This enables you to set the routing condition, or specify the value type, as a regular expression. Objects are routed to collections if their values, or object\_name attributes, match that expression as shown in the following example:

```
<properties>
  <property name="routing-field-xpath" value="dmftmetadata//object_name" />
</properties>
```

## Customizing a routing class

If xPlore routing classes do not meet your requirements, you can implement a customized routing class based on the following guidelines:

- The customized routing class must extend *com.emc.documentum.core.fulltext.indexserver.core.index.routing.FtAbstractCollectionRouting*
- The class must implement the method *public String getCustomCollection(IFtIndexRequest request, IFTDomain domain) throws IndexRoutingException*
- The configuration must be similar to the xPlore predefined routing class
- If the routing-class is used to configure properties, use the incoming parameter's *config* object in the constructor *public FtAbstractCollectionRouting(ICollectionRoutingConfig config)*. This includes a map of *<string, string>* for properties, and a *FtRoutingRule* list for the configured rule list.

- The incoming DOM will be in the request object `request.getDocument().getContentToIndex()`. Use DOM structure navigation or XPath evaluation to get the value of the relevant field. The new custom routing class and its dependent jar files should be added to dsearch.war/WEB-INF/lib.

### 7.5.8 Attaching and detaching a collection



**Note:** To avoid corruption, do not detach or attach a collection during index rebuild.

Attach and detach are required for a restore operation. See “[Performing an offline restore](#)” on page 218.

### 7.5.9 Moving a collection

If you are moving a collection to another xPlore instance, choose the collection and click **Configuration**. Change **Binding instance**.

You can create additional collections for faster ingestion, and then move it to become a subcollection of another collection after ingestion has completed. When you move it as a subcollection, search performance is improved.

If a collection meets the following requirements, you can move it to become a subcollection:

- Facet compression was disabled for all facets before the documents were indexed.
  - The collection is not itself a subcollection (does not have a parent collection).
  - The collection does not have subcollections.
  - The collection has the same category and index definitions (in `indexserverconfig.xml`) as the new parent.
  - xPlore enforces additional restrictions after the collection has been moved. For information on subcollection restrictions, see “[Uses of subcollections](#)” on page 197.
1. Choose the collection and click **Move**.
  2. In the **Move to** dialog box, select a target collection. This collection can be a subcollection or a top-level collection.

### 7.5.10 Moving a collection to a new drive

You can move one or more collections when you are changing disks or a drive is full. The following procedure assumes that you do not move the xplore\_home/config directory.

1. Stop the index agent and xPlore instances.
2. Back up the xplore\_home /data and xplore\_home /config directories.
3. Copy the *xplore\_home*/data folder to the new path. This location must be accessible and writeable by all xPlore instances. Repeat this step if you created one or more storage locations using a different path.
4. Update the old location to the new location in the following files on the primary instance host:
  - indexserverconfig.xml. For example:
 

```
<storage-location status="not_full" quota_in_MB="10"
path="C:/xPlore_1/data-new" name="default"/>
```
  - XhiveDatabase.bootstrap. Update the path of each segment to the new path. For example:
 

```
<segment reserved="false" library-id="0" library-path=
"/Repository1/dsearch/Data/default" usable="true"
usage="detachable_root" state="read_write" version="1"
temp="false" id="Repository1#dsearch#Data#default">
  <file id="12" path="C:/xPlore_1/data-new|Repository1|default|
xhiveDb-Repository1#dsearch#Data#default-0.XhiveDatabase.DB"/>
  <binding-server name="primary" />
</segment>
```
5. On all instances, change the path in indexserver-bootstrap.properties to match the new bootstrap location. For example:
 

```
xhive-data-directory=C:/xPlore_1/data-new
```
6. Delete the WildFly cache (/work folders) from the index agent and primary instance web applications. Also delete WildFly cache folders for the secondary instances. The path of the /work folder is *<xPlore\_home>/<wildfly\_version>/server/DctmServer\_<InstanceName>/work*.
7. Start the xPlore instances.
8. Start the index agent.
9. Import a test document and search for it using xPlore administrator. Also, search for a document that has already been indexed.

### 7.5.11 Creating a storage location

xPlore stores data and indexes in an xDB database. The index for each collection can be routed to a specific storage location. The storage location is not specific to one instance, it must be shared among instances.

After you create a storage location, you can select it when you create a domain or a new collection. A collection can be stored in a location different from other collections in the domain. After you choose the location, you must back up and restore to the same location. If the disk is full, set the collection state to `update_and_search` and create a collection in a new storage location.

1. In xPlore administrator, choose **System Overview** in the tree.
2. Click **Global Configuration** and then choose the **Storage Location** tab.
3. Click **Add Storage**. Enter a name and path and save the storage location. The storage location is created with unlimited size.

### 7.5.12 Monitoring merges of index data

You can monitor the merge of index data from in-memory to the xDB database. When a collection is merging, you see a **Merging** icon next to the collection in the Data Management view of the collection. Because merging can slow requests, you can stop or start the merge from the UI. If merging is in progress, click **Stop Merging**. If you want to initiate merging, click **Start Merging**.

For merging commands, see “[Final merge CLIs](#)” on page 236. “[Managing sub-index merges](#)” on page 377 describes the types of merges and how to manage final merges.

### 7.5.13 Querying a collection

1. Choose a collection in the Data Management tree.
2. In the right pane, click **Execute XQuery**.
3. Check **Get query debug** to debug your query. The query optimizer is for use by OpenText Global Technical Services.

To route queries to a specific collection, see “[Routing a query to a specific collection](#)” on page 317.

## 7.6 Rebuilding indexes

You can rebuild indexes either by refeeding data from the original source, or perform an online index rebuild, depending on your use case. Starting from xPlore 1.5, index rebuilding is performed on collection bindings instead of the primary node (previous behavior). This change improves the index scalability and reduces load on the primary node.

- Data refeed

You use the data refeed approach to rebuild indexes in the following scenarios:

- Database corruption involves index, data pages, and redo log
- You want to restore from some massive failure without any backup
- There are some changes to the system that significantly impact text extraction
- It is OK for the system to be out of service when you rebuild the index
- You changed the data model to add indexable attributes for an object type.

- Online index rebuild

You perform online index rebuild in the following scenarios:

- Database corruption involves only Lucene (multi-path index)
- There have been schema changes to the underlying index such as improved data typing
- Downtime is highly undesirable
- You added a stop words list.
- You added a new facet.
- You want to strip out accents from words in the index.
- You want to use index agent or DFC filters, and the objects have already been indexed.

### 7.6.1 Rebuilding indexes through data refeed

You must remove all indexes before performing data refeed.

To rebuild indexes by refeeding data:

1. Shut down all xPlore instances.
2. If you only have a single domain and it is corrupted, do the following:
  1. Delete everything under `<xPlore_home>/data`.
  2. Delete everything under `<xPlore_home>/config` except the file `indexserverconfig.xml`.
3. Start xPlore instances.

4. Refeed the documents to launch a full reindexing: in the index agent UI, select **Start new reindexing operation**.

## 7.6.2 Configuring online index rebuilds

Set the following parameters in `indexserverconfig.xml` to configure online index rebuilds:

- `rebuild-index-thread-number`

Specify the number of threads to use for online index rebuild. Default: 1 (single-thread mode).

Set this value to greater than 1 to enable the parallel multi-thread online index rebuild feature, which is noticeably faster than single-thread reindexing, but at the cost of more I/O and disk usage, and slower querying performance.

- `rebuild-index-task-retry-time`

The number of times xPlore retries online index rebuild when errors occur. Default: 3.

- `rebuild-index-enable-final-merge`

Whether to enable the final merge during online index rebuild:

- true: During online index rebuild, the final merge still runs as scheduled. Default: true.

- false: Final merges are disabled during online index rebuilds. When the online index rebuild is complete, the system automatically triggers a final merge on the reindexed collection.

Since the final merge is very I/O intensive and may cause noticeable performance drop in both indexing and query performance when running, it is recommended that you set this value to false to prevent the final merge from running during online index rebuilds. After rebuilding is completed, the final merge is executed immediately to ensure query performance is optimized.

Restart xPlore for the settings to take effect.

## 7.6.3 Performing an online index rebuild

A collection must be online and in the read/write state to rebuild the collection. You can perform normal ingestion and search during index rebuild.



**Note:** Do not perform the following operations during rebuild: Heavy ingestion load, backup or restore, attach or detach a domain or collection, create collection, or check consistency. If necessary, you can suspend a long rebuild and then resume it at a later time.

Rebuild a collection index in the xPlore administrator UI. Custom routing is not applied when you rebuild an index. To apply custom routing, refeed the documents. See “[Deleting a collection and recreating indexes](#)” on page 207.

To perform an online index rebuild:

1. Select a domain or collection in the **Data Management** tree. If you are rebuilding the entire domain, open each collection to perform the next step.
2. In the right pane, click **Rebuild Index** to start the rebuild. While it is in progress, you can suspend the rebuild and resume it at a later time, or you can cancel the rebuild.

The index is rebuilt in the background and then put into place when the rebuild is complete.

3. Reingest large documents. At the original ingestion, some documents are not embedded in dftxml because the XML content exceeds the value of the file-limit attribute on the xml-content element in indexserverconfig.xml. The index rebuild process generates a list of object IDs for these documents. Use this list in the next step. The list is located in `xplore_home/data/domain_name/collection_name/index_name/ids.txt`.

For example:

`C:/xplore/data/mydomain/default/dmftdoc_2er90/ids.txt`

4. In the index agent UI, provide the path to the list of object IDs. See “[Indexing documents in normal mode](#)” on page 103.
5. After the index is rebuilt, run *ftintegrity* or the *State of Index* job in Documentum Server. See “[Using ftintegrity](#)” on page 94 or “[Running the state of index job](#)” on page 98.

## 7.6.4 Deleting a collection and recreating indexes

Normally, you do not need to delete an existing collection and refeed a new one to recreate indexes. Changes such as defining new facets and configuring the wildcard optimization of an attribute do not require a refeed. All you need to do is to perform an online index rebuild.

However, in some cases, such as when the object type definition in Documentum has changed in a manner that invalidates the way it was stored in xPlore before, refeeding a new collection is more efficient than deleting everything from the existing collection and then refeeding it. For example, a Documentum object attribute named `customer_date` was originally typed as an integer as a numeric timestamp but was later retyped as a date format. xPlore stored the objects originally as integer, so the objects need to be removed and reloaded. In such cases, you need to create and refeed a new collection.

1. Remove the problematic collections:
  - a. Navigate to the domain in xPlore administrator left panel.
  - b. Click the X next to the collection name to delete it.
2. Modify the index definition, if necessary. After modifying `indexserverconfig.xml`, you must restart xPlore for the changes to take effect.

3. Create a collection with the same name as the one you deleted.
4. Refeed the documents to launch a full reindexing: in the index agent UI, select **Start new reindexing operation**.

If custom routing is defined, it is applied. Otherwise, default routing is applied.

## 7.7 Checking xDB statistics

A verbose command, `statistics-ls`, lists all fields and tokens in the collection. You can use this command to check whether certain Documentum attributes were tokenized.

Follow these steps to check xDB statistics:

1. Open a command window and navigate to `xplore_home/dsearch/xhive/admin`.
2. Launch `XHCommand.bat` (Windows) or `XHCommand` (Linux).
3. Enter the `xDB statistics-ls` command with relevant options.

Syntax:

```
statistics-ls [options] path
```

The `-d` argument (`xhivedb`) is the same for all xPlore installations. The `path` argument is the path within xDB to your collection, which you can check in the XHAdmin tool. The output file is large for most collections. Redirect output to a file using the `-o` option. (The file must exist before you run the command.) For example:

```
statistics-ls -d xhivedb -o C:/temp/DefaultCollStats.txt  
/TechPubsGlobal/dsearch/Data/default --lucene-params --details
```

The output gives the status of each collection (Merge State). For example:

```
LibPath=/PERFXCP1/dsearch/Data/default IndexName={dmftdoc}  
ActiveSegment=EI-8b38b821-4e29-42b2-9fe0-8e6c82764a6b-211106232537097-luceneblobs-1  
EntryName=LI-3bb3483d-38c9-4d14-8a90-5a13a9a19717  
MergeState=NONE isFinalIndex=FINAL  
LastMergeTime=12/09/2012-07:31:11 MinLSN=0 MinLSN=0  
LibPath=/PERFXCP1/dsearch/Data/default IndexName={dmftdoc}  
ActiveSegment=EI-8b38b821-4e29-42b2-9fe0-8e6c82764a6b-211106232537097-luceneblobs-1  
EntryName=LI-2ea1578b-0d82-496a-9c81-ee15502b3cbe  
MergeState=NONE isFinalIndex=NOT-FINAL  
LastMergeTime=14/09/2012-10:15:48 MinLSN=786124  
MinLSN=485357881901
```

### Other statistics

You can check other statistics such as returnable fields, size of index, and number of documents. The `statistics` command has the same arguments as `statistics-ls`.

For example:

```
statistics --docs-num -d xhivedb /TechPubsGlobal/dsearch/Data/default dmftdoc
```

Statistics options:

- `--lucene-sz`: Size of Lucene fields (.fdt), Index to fields (.fdx), and total size of Lucene index in bytes (all).

- `--lucene-rf`: Statistics of returnable fields (configured in `indexserverconfig.xml`). Includes total count of path mapping and value mapping and compression mapping consistency.
- `--lucene-list`: Shows whether each index is final.
- `--lucene-params`: Lists the xDB parameters set in `xDB.properties`.
- `--docs-num`: Displays the number of documents in the collection. This value should match the number displayed for a collection in xPlore administrator.

## 7.8 Troubleshooting data management

### Auditing collection operations

You can audit operations on collections. Auditing is enabled by default. To enable or disable auditing, open **Global Configuration** in the xPlore administrator left pane. Select the **Auditing** tab and check **admin** to enable or disable data management auditing.

For information on configuring the audit record, see [“Configuring the audit record” on page 48](#).

When auditing is enabled, the following operations on a collection are recorded:

- Create and delete collection
- Add, remove, or change binding
- Create, rebuild start and end, or delete index
- Attach or detach
- Adopt collection start and end
- Backup start and end

You can view a report based on the admin audit records. Go to **Diagnostic and Utilities > Reports** and choose **Audit records for admin component**. You can filter by date.

### Force restart of xDB

If xDB fails to start up, you can force a start.

1. Edit `indexserver-bootstrap.properties` in the `WEB-INF/classes` directory of the application server instance, for example, `xplore_home/<wildfly_version>/server/DctmServer_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes`.
2. Set the value of `force-restart-xdb` in `indexserver-bootstrap.properties` to true. Restart the xPlore instance.
3. If this property does not exist in `indexserver-bootstrap.properties`, add the following line:

```
force-restart-xdb=true
```



**Note:** This property will be removed after restart.

Do not remove segments from xDB unless they are orphan segments. (See “[Orphaned segments CLIs](#)” on page 233.) If you do, your backups cannot be restored.

## 7.9 xDB repair commands

You can troubleshoot the following issues by stopping xPlore and starting the xDB server in repair mode. Perform the following steps:

1. Stop all xPlore instances and any xPlore-related Java processes.
2. Open a command-line window and navigate to *xplore\_home/dsearch/xhive/admin*.
3. Enter the following command: XHCommand.bat (Windows) or XHCommand (Linux). You see the xdb prompt.
4. Start the xDB server with the *run-server-repair* command. The xDB port is recorded in the xDB bootstrap file. Syntax:

```
run-server-repair --port <xDBport> -f <path_to_xDB_bootstrap>
```

For example:

```
xdb>run-server-repair --port 9330 -f C:/xPlore/config/XhiveDatabase.bootstrap
```

When the xDB server starts successfully, you see a message like the following:

```
xDB 10_2@1448404 server listening at xhive://0.0.0.0:9330
```

5. Open a new command window with XHCommand to enter a repair command.
6. When finished, enter the command to stop the repair server mode:

```
stop-server --nodename primary
```

### Command arguments

To list all available commands, enter *help* at the xdb prompt. To get arguments for any of the commands, enter *<command> help*, for example:

```
xdb>help repair-merge
```

The *path\_to\_index* argument in repair commands is a library path, not a file system path. For example: *repository\_name/dsearch/Data/default*. The *index\_name* value is *dmftdoc*. *dmftdoc* is the multi-path index for Documentum repositories.

### Force a final merge to improve query performance

The final merge is executed in a certain interval in the multi-path index. Query performance improves after final merge. By default, final merge is executed every 4

hours. In the xDB server repair mode, you can trigger a final merge manually using the following Syntax:

```
xdb>repair-merge -d xhivedb <path_to_index> <index_name>
```

For example:

```
repair-merge -d xhivedb LH1/dsearch/Data/default dmftdoc --final
```

A successful merge is reported like the following:

```
Repairing options: {REPAIR_MERGE_FINAL_SUB_INDEXES={}}
Index repair report:

index entry : LI-303cf8ed-d0f4-454b-b54f-dd48817cffd8 has
been merged into entry:
LI-abf1d2de-1783-4ce7-a013-6cdf9e871bed

index entry :
LI-a7f66aa5-a729-4774-8b97-0950b0703f3c has
been merged
into entry:
LI-abf1d2de-1783-4ce7-a013-6cdf9e871bed
```

## Index corruption: Repair segments

For an INDEX\_CORRUPTION error, run the *repair-segments* command. This repair takes long time for a large index. Back up the index before you run it. The command is not guaranteed to fix all data corruptions. Syntax:

```
repair-segments -d xhivedb <path_to_index> dmftdoc
```

For example:

```
repair-segments -d xhivedb LH1/dsearch/Data/default dmftdoc
```

If all segments pass repair check, you see the following:

```
Repairing options: {REPAIR_LUCENE_INDEXES_SEGMENTS={}}
Index repair report:
Index LI-a7f66aa5-a729-4774-8b97-0950b0703f3c is ok
Index LI-303cf8ed-d0f4-454b-b54f-dd48817cffd8 is ok
```

## Object dead: Fix broken blacklists

For a OBJECT\_DEAD error, run the repair-blacklists command. Syntax:

```
repair-blacklists -d xhivedb <path_to_index> dmftdoc --check-dups
--repair-index
```

For example:

```
repair-blacklists -d xhivedb LH1/dsearch/Data/default dmftdoc --
check-dups --repair-index
```

A successful check is like the following:

```
Repairing options:
{REPAIR_LUCENE_CHECK_AND_FIX_BLACKLISTS={
REPAIR_INDEX_LIBRARY=LH1/dsearch/Data/default,
REPAIR_PERFORM_FIX_ACTIVITIES=false,
REPAIR_INDEX_CHE
```

```
CK_DUPLICATE_NODES_CONSISTENCY=true}}  
Index repair report:  
  
The black lists are :  
  
processing index  
C:\xPlore\data\LH1\default\lucene-index\dmftdoc\EI-d1071aae-a76c-46ee-825e-6955c  
313954b  
  
processing index  
C:\xPlore\data\LH1\default\lucene-index\dmftdoc\EI-d1071aae-a76c-46ee-825e-6955c  
313954b  
  
total docs processed = 1  
total checked blacklisted objects = 0  
total unaccounted for blacklisted objects = 0  
total duplicate entries found = 0  
total intranode  
dups found = 0
```

# Chapter 8

## Backup and Restore

### 8.1 About backup

Back up a collection, domain, or xPlore federation after you make xPlore environment changes: Adding or deleting a collection, or changing a collection binding. If you do not back up, then restoring the domain or xPlore federation puts the system in an inconsistent state. Perform all your anticipated configuration changes before performing a full federation backup.

You can back up an xPlore federation, domain, or collection using xPlore administrator or use your preferred volume-based or file-based backup technologies. The *OpenText Documentum xPlore Installation Guide* describes high availability and disaster recovery planning.

You can use external automatic backup products like OpenText Networker. All backup and restore commands are available as command-line interfaces (CLI) for scripting. See the chapter Automated Utilities (CLI).

If the disk is full, set the collection state to `update_and_search` and create a collection in a new storage location.



**Note:** If you remove segments from xDB, your backups cannot be restored.

### Backup and restore consistency check

Perform a consistency check before backup and after restore.

Select **Data Management** in xPlore Administrator and then choose **Check DB Consistency**. This check determines whether there are any corrupted or missing files such as configuration files or Lucene indexes. Lucene indexes are checked to see whether they are consistent with the xDB records: Tree segments, xDB page owners, and xDB DOM nodes. For a tool that checks consistency between the index and the tracking database, see “[Running the standalone data consistency checker](#)” on page 185.

### Order of backup and restore in Documentum systems

The backup and restore date for xPlore must be at a point in time earlier than the backup of repository content and metadata. Use `ftintegrity` after the restore to detect additional files that need indexing. (These files or ACLs were added or updated after the xPlore backup.) See “[Using ftintegrity](#)” on page 94.

### Backup state: Hot, warm, and cold

You can back up the entire xPlore federation while you continue to search and index (hot backup). You can back up a domain (Documentum repository index) or

collection with indexing suspended (warm backup). When you back up using xPlore administrator, the state is set to search\_only for domain and collection backup and reverted after the backup completes.

## Backup Methods

xPlore supports the following backup approaches.

- Native xDB backups: These backups are performed through xPlore administrator. You can back up hot (while indexing), warm (search only), or cold (offline). See [“Performing a native xDB backup in xPlore Administrator” on page 217](#).
- File-based backups: Back up the xPlore federation directory *xplore\_home*/data, *xplore\_home*/config, and /wal files. Backup is warm (search only) or cold (offline). Incremental file-based backups are not recommended, since most files are touched when they are opened. In addition, Windows file-based backup software requires exclusive access to a file during backup and thus requires a cold backup.
- Volume-based (snapshot) backups: Backup is warm (search only) or cold (offline). Can be incremental or full backup of disk blocks. Volume-based backups require a third-party product such as EMC Timefinder.

You can use the CLI tool to perform scripted backup for the above-mentioned backups. See [“Scripted backup” on page 229](#).

A snapshot, which is a type of point-in-time backup, is a backup of the changes since the last snapshot. A copy-on-write snapshot is a differential copy of the changes that are made every time new data is written or existing data is updated. In a split-mirror snapshot, the mirroring process is stopped and a full copy of the entire volume is created. A copy-on-write snapshot uses less disk space than a split-mirror snapshot but requires more processing overhead.

## Backup combinations

Periodic full backups are recommended.

**Table 8-1: Backup scenarios**

Level	Backup state	Backup method	Backup scope
collection	warm	Native	full
domain	warm	Native	full
federation	warm or hot cold or warm cold or warm	Native volume* file*	full full full

## Synchronous and asynchronous replication

Replication can be asynchronous or synchronous. In synchronous replication, the primary system must receive a confirmation from the remote system that the data was successfully received. So, the data is always in sync, but performance can be slow because of large data volume or the long distance between systems. In asynchronous replication, the primary system does not wait for the remote system to confirm that the data was received. Performance is faster than synchronous replication, but the data is not always in sync. There is also the potential for data loss, so consistency procedures must be run upon recovery.

Some asynchronous replication technologies support consistency groups. For these applications, the data is in sync. You define a group of storage areas for interdependent applications, such as the Documentum Server, RDBMS, full-text indexes, to be coordinated. Furthermore, that consistency group monitors all disks that are assigned to it (as well as the I/O writes to these disks) to ensure write-order fidelity across these disks. EMC Symmetrix Data Remote Facility/Asynchronous (SDRF/A) supports consistency groups.

## Backup risk and space planning

In any backup strategy, there is a certain amount of inherent risk. For example, backup files can be corrupted or lost.

To mitigate risk:

- Completely test your procedures to make sure that you can recover and make your xPlore installation operational within the required RTO and RPO. (An operational deployment is able to index and return query results.)
- A best practice is to test periodically your backups (for example, by performing a restore in a test environment) to confirm that the backup process is working properly.

As the number of backups increase, the space required to store them will also increase. Therefore, you should consider how long you are required to retain your backups (which is also referred to as a *data retention period*). Plan for the appropriate amount of storage space.

## 8.2 About restore

All restore operations are performed offline. If you performed a hot backup using xPlore administrator, the backup file is restored to the point at which backup began.

Each xPlore instance owns the index for one or more domains or collections, and a transaction log. If there are multiple instances, one instance can own part of the index for a domain or collection.



**Note:** You cannot restore the backup of an earlier version of xPlore, because the xDB version has changed. Back up your installation immediately after upgrading xPlore.

### Backup and restore consistency check

Perform a consistency check before backup and after restore.

Select **Data Management** in xPlore Administrator and then choose **Check DB Consistency**. This check determines whether there are any corrupted or missing files such as configuration files or Lucene indexes. Lucene indexes are checked to see whether they are consistent with the xDB records: Tree segments, xDB page owners, and xDB DOM nodes. For a tool that checks consistency between the index and the tracking database, see “[Running the standalone data consistency checker](#)” on page 185.

### Scripted restore

Use the CLI for scripted restore of a federation, collection, domain. See the chapter “[Automated Utilities \(CLI\)](#)”. xPlore supports offline restore only. xPlore instances must be shut down to restore a collection or an xPlore federation.

### Order of backup and restore in Documentum systems

The backup and restore date for xPlore must be at a point in time earlier than the backup of repository content and metadata. Use *ftintegrity* after the restore to detect additional files that need indexing. (These files or ACLs were added or updated after the xPlore backup.) See “[Using ftintegrity](#)” on page 94

After you back up an xPlore federation, you can either restore it to the same xPlore deployment from which the backup is created or restore it to a different xPlore deployment.

For information about local restore, see “[Performing an offline restore](#)” on page 218.

For information about remote restore, see “[Performing a remote restore](#)” on page 219.

## 8.3 Performing a native xDB backup in xPlore Administrator

When you perform a native xDB backup in xPlore Administrator, xDB can be in hot, warm, or cold state:

- Hot backup: You do not need to stop any services. Indexing and search continues during the backup process.
- Warm backup: Suspend the indexing service on each instance that the collection is bound to. To do so, choose the service under the instance and click **Operations**; then click **Disable** to suspend the service.
- Cold backup: Suspend both the indexing and search service on each instance that the collection is bound to. To do so, choose the service under the instance and click **Operations**; then click **Disable** to suspend the service.

In xPlore Administrator, you can back up the federation, a domain, or a collection. The following steps



**Note:** Backup is not available for a subcollection. Back up the parent and all its subcollections.

To perform a hot backup or warm backup, use the following procedure. It is not applicable for cold backups, since xPlore instances are suspended.

1. Select the item you want to back up:
  - Federation: Click **Data Management**.
  - Domain or collection: Under **Data Management**, click the domain or collection you want to back up.
2. Click **Backup**.
3. Specify a backup location or choose the default one.  
For federation backup, choose **Full backup**.
4. When backup is complete, a message is displayed showing where backup files are located.
5. Back up your jars or DLLs for custom content processing plugins or annotators.

## 8.4 Performing an offline restore

xPlore instances must be shut down to restore a collection, domain, or xPlore federation.

This procedure assumes that no system changes (new or deleted collections, changed bindings) have occurred since backup. (Perform a full federation backup every time you make configuration changes to the xPlore environment.)

If you are restoring a federation and a collection that was added after the federation backup, do the following:

1. Restore the federation.
2. Start up and shut down xPlore.
3. Restore the collection.

The following instructions include some non-scripted steps in xPlore administrator.

1. Federation only: Shut down all xPlore instances and clean up all existing data files.
  - Delete everything under *xplore\_home*/data.
  - Delete everything under *xplore\_home*/config.
2. Force-detach the domain or collection:
  1. Collection only : Set the collection state to off\_line using xPlore administrator. Choose the collection and click **Configuration**.
  2. Domain or collection: Detach the domain or collection using xPlore administrator.



**Note:** Force-detach corrupts the domain or collection.

3. Domain only: Generate the orphaned segment list. Use the CLI *listOrphanedSegments*. See “[Orphaned segments CLIs](#)” on page 233.
4. Stop all xPlore instances.
5. Run the restore CLI. See “[Scripted federation restore](#)” on page 230, “[Scripted domain restore](#)” on page 230, or “[Scripted collection restore](#)” on page 231.
6. Start all xPlore instances. No further steps are needed for federation restore. Do the following steps for domain or collection restore.
7. Domain only: If orphaned segments are reported before restore, run the CLI *purgeOrphanedSegments*. If an orphaned segment file is not specified, the orphaned segment IDs are read from *stdin*.
8. Force-attach the domain or collection using xPlore administrator.

9. Perform a consistency check and test search. Select **Data Management** in xPlore Administrator and then choose **Check DB Consistency**.
10. Run the ACL and group replication script to update any security changes since the backup. See ["Manually updating security" on page 64](#).
11. Run *ftintegrity*. For the start date argument, use the date of the last backup, and for the end date use the current date. See ["Using ftintegrity" on page 94](#).

For automated (scripted) restore, see ["Scripted federation restore" on page 230](#), ["Scripted domain restore" on page 230](#), or ["Scripted collection restore" on page 231](#).

For remote restore, see ["Performing a remote restore" on page 219](#).

## 8.5 Performing a remote restore

A remote restore restores a federation backup of an xPlore deployment to another xPlore deployment.

A remote restore has the following prerequisites:

- The administrator password for the target deployment (to which to restore the backup) must be the same as the one for the source deployment (of which the backup is created).
- The source deployment and the target deployment must have the same number of instances with identical names.
- The source deployment and the target deployment must have the same number of storage areas with identical names.

To perform a remote restore,

1. Make sure `indexserverconfig.xml` exists in the target xPlore deployment. Make sure you do not delete this file.
2. Copy the federation backup from the source deployment to the host where the primary instance of the target deployment resides.
3. Perform an offline restore on the host where the target deployment resides. Execute the following command to find out the syntax of the remote restore command.
  - Windows: `xplore.bat restoreFederation /?`
  - Linux: `./xplore.sh restoreFederation /?`

## 8.6 Performing a file- or volume-based (snapshot) backup and restore

Data files in the backup must be on a single volume.

This procedure assumes that no system changes (new or deleted collections, changed bindings) have occurred since backup. Perform all your anticipated environment changes before backup. Make sure that you have sufficient disk space for the backup and for temporary space (twice the present index size).



**Note:** Do not mix CLI commands (suspend or resume disk writes) with native xPlore backup in xPlore administrator.

1. Set all domains in the xPlore federation to the *read\_only* state using the CLI. (Do not use native xPlore to set state.) See “[Collection and domain state CLIs](#)” on page 234.
2. Suspend ingestion for backup or restore. Search is still enabled.
  - a. Navigate to *xplore\_home/dsearch/xhive/admin*.
  - b. Launch the command-line tool with the following command. You supply the administrator password (same as xPlore administrator).

```
XHCommand suspend-diskwrites
```
3. Use your third-party backup software to back up or restore.
4. Resume xDB with the following command:

```
XHCommand suspend-diskwrites --resume
```
5. Set all backed up domains to the *reset* state and then turn on indexing. (This state is not displayed in xPlore administrator and is used only for the backup and restore utilities.) Use the script in “[Collection and domain state CLIs](#)” on page 234.

## 8.7 Troubleshooting backup and restore

### Volume-based backup of domain or collection is not supported

Volume-based backup requires change of a domain or collection state to read-only. As a result, you can back up an xPlore federation with volume-based backup. You cannot use volume-based backups for domains or collections.

### Federation and collection restore procedure not followed

Restoring a federation and then a collection that was backed up after the federation can lead to data corruption. Start and then stop xPlore after you restore the federation. Then it is safe to restore the collection.

Perform the following steps:

1. Restore the federation.
2. Start all xPlore instances.
3. Stop all xPlore instances.
4. Restore the collection.

## CLI troubleshooting

If a CLI does not execute correctly, check the following:

- The output message may describe the source of the error.
- Check whether the host and port are set correctly in *xplore\_home/dsearch/admin/xplore.properties*.
- Check the CLI syntax.
  - Linux requires double quotes before the command name and after the entire command and arguments.
  - Separate each parameter by a comma.
  - Do not put Boolean or null arguments in quotation marks. For example:

```
xplore "backupFederation null,false,null"
```
- Check whether the primary instance is running.

```
http://instancename:port/dsearch
```
- Check whether the JMX MBean server is started. Open jconsole in *xplore\_home/java64/java\_version/bin* and specify **Remote Process** with the value *service:jmx:rmi://jndi/rmi://myhost:9331/dsearch*. (9331 is the default JMX port. If you have specified a base port for the primary instance that is not 9300, add 31 to your base port.) If jconsole opens, the JMX layer is OK.
- Check the Admin web service. Open a browser with the following link. If the XML schema is shown the web service layer is operative.

```
http://instancename:port/dsearch/ESSAdminWebService?wsdl
```
- Check *dsearch.log* in *xplore\_home/<wildfly\_version>/server/instance\_name/logs* for a CLI-related message.

## 8.8 Handling data corruption

### 8.8.1 Detecting data corruption

You can detect data corruption in the following ways:

- An XhiveDataCorruptionException is reported in the xPlore server log.
- Run the consistency checker on the xPlore federation. See “[Domain and collection menu actions](#)” on page 191.
- The primary instance cannot start up.

### 8.8.2 Handling a corrupt domain

1. Use xPlore administrator to set the domain mode to *maintenance*. You can also use the CLI setDomainMode. See “[Domain mode CLIs](#)” on page 234.

In maintenance mode, the following restrictions are applied:

- The only allowed operations are repair and consistency check.
- Only queries from xPlore administrator are evaluated.
- Queries from a Documentum client are tried as NOFTDQL in the Documentum Server. xPlore does not process them.

2. Detach the corrupted domain.
3. Restore the corrupted domain from backup. See “[Performing an offline restore](#)” on page 218.

When xPlore is restarted, the domain mode is always set to normal (*maintenance* mode is not persisted to disk).

### 8.8.3 Repairing a corrupted index

The console on startup reports XhiveException: LUCENE\_INDEX\_CORRUPTED. The xPlore primary instance may not start up.

1. If the xPlore primary instance does not start, do the following:
  - a. Force the server to start up: Set the value of *force-restart-xdb* in indexserver-bootstrap.properties to true. The corrupt collection is marked *off\_line* and updates are ignored. Restore the corrupted domain or collection from a backup.
  - b. If the xPlore server starts up, choose the collection in the tree, click **Configuration**, and then set the state to *off\_line*. The offending collection and its index are marked as unusable and updates are ignored.
  - c. Repair or restore from backup the corrupted collection or log. Continue with the next step.

2. Start the xDB admin tool XHAdmin.bat (Windows) or XHAdmin (Linux) and drill down to the library that was reported as corrupted.
3. Right-click and choose **Library management > Check library consistency**. If the consistency check fails, perform the xDB command repair-segments.
  - a. Open a command window and navigate to `<xPlore_home>/dsearch/xhive/admin`.
  - b. Launch XHAdmin.bat (Windows) or XHAdmin (Linux).
  - c. Enter the following xDB command:
 

```
xdb>run-server-repair -f <path to XhiveDatabase.bootstrap file> -port <xDB port>
```
  - d. Launch XHAdmin.bat (Windows) or XHAdmin (Linux) in another window.
  - e. Enter the following xDB command:
 

```
xdb>repair-segments -d <database> -p <path> <target>
```

`<database>` is the name of the xDB database, generally `xhivedb`.  
`<path>` is the full path to the library (collection) containing the index. Get the path in the xPlore administrator collection view.  
`<target>`(optional) is the Lucene index name reported in the error, such as `dmftdoc`.

For example:

```
repair-segments -d xhivedb -p ot101! /xPlore/dsearch/Data/default dmftdoc
```
4. Restart all xPlore instances.`lmpi-finalmerge-day-blackout = 8:15`
5. Run the consistency checker again.
  - If the consistency check passes, the system is usable.
  - If the consistency check fails, rebuild the index.

#### 8.8.4 Cleaning and rebuilding the index

Use this procedure for the following use cases:

- Data is corrupted on disk and there is no backup
  - You change collections after backup and see errors when you try to restore. You see errors if you added or deleted a collection or changed collection binding after backup.
1. If the xPlore system does not restart, try a force restart. Set the value of `force-restart-xdb` in `indexserver-bootstrap.properties` to true. (This file is located in the WEB-INF/classes directory of the application server instance, for example, `C:\xPlore\wildfly9.0.1\server\DCtmServer_PrimaryDsearch\deployments\dsearch.war\WEB-INF\classes`.)

2. Restart the xPlore instance. If startup still reports errors, go to the next step to clean corrupted data.
3. Shut down all xPlore instances.
4. Delete everything under *xplore\_home*/data.
5. Delete everything under *xplore\_home*/config except the file indexserverconfig.xml.
6. Start xPlore instances.
7. Choose one:
  - You have a backup. Restore it using the procedure described in “[Performing an offline restore](#)” on page 218. Recreate any collection that was added after backup.
  - You do not have a backup. Refeed all documents. (Use migration instructions in *OpenText Documentum xPlore Installation Guide*.)

### 8.8.5 Dead objects

For the XhiveException: OBJECT\_DEAD.

1. In xPlore administrator, make sure that all collections have a state of index\_and\_search.
2. Stop xPlore instances.
3. Start xDB in repair mode.
  - a. Change the memory in xdb.properties for all nodes. This file is located in <*xplore\_home*>/<*wildfly\_version*>/server/<*instance\_name*>/deployments/dsearch.war/WEB-INF/classes/xdb.properties. This change can remain after repair.  
XHIVE\_MAX\_MEMORY=1536M
  - b. Start each instance in repair mode: Open a shell, go to the directory *xplore\_home*/dsearch/xhive/admin/, run XHCommand.bat (Windows) or XHCommand (Linux), and input the instance name, port, and path to the bootstrap file on the host.

For example:

```
run-server-repair --address 0.0.0.0 --port 9330 --nodename primary  
-f %XPLORE%/config/XhiveDatabase.bootstrap
```

4. Input the following xhcommand, specifying the domain, collection name, and parameter. The repair command can take a long time if the index is large.

To scan without removing dead objects, remove the option '--repair-index':

```
repair-blacklists -d xhivedb /%DOMAIN%/dsearch/Data/%COLLECTION% dmftdoc  
--check-dups --repair-index
```

5. Stop repair mode using xhcommand. For example:

```
stop-server --nodename primary
```

6. Start all xPlore instances.

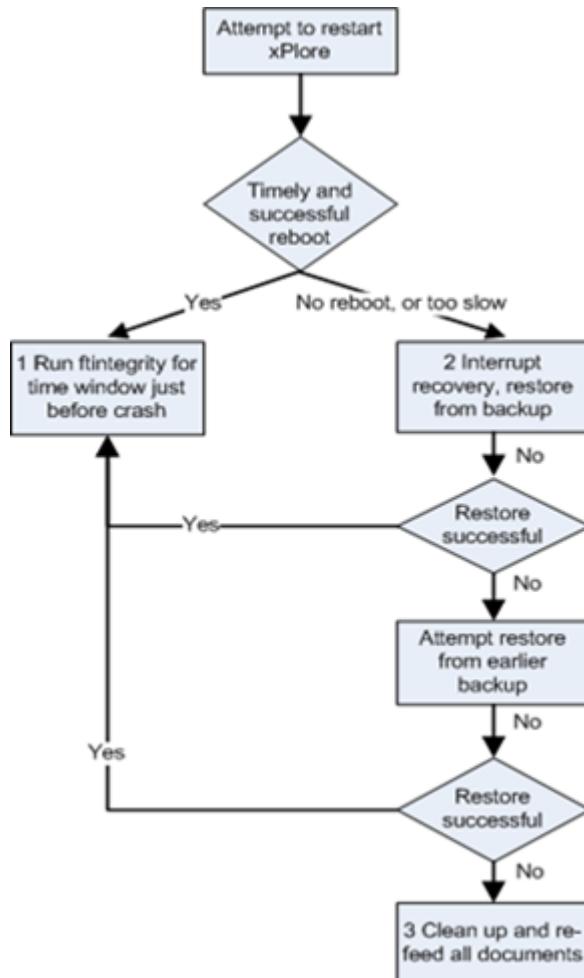
Check the standard output from step 4. You see a summary of Total dead objects and Total potential impacted normal objects and a detailed report. For example:

```
("total docs processed =  
("total checked blacklisted objects = "  
("total unaccounted for blacklisted objects = "  
("total duplicate entries found = ");  
("total intranode dups found = "
```

If Total potential impacted normal objects is not 0, a file is generated with the following name convention: %DOMAIN%#dsearch#Data#%COLLECTION %\_objects\_2012-03-12-21-02-06. Resubmit this file using the index agent UI.

1. Log in to the index agent UI.
2. Choose **Object File**.
3. Browse to the file and choose **Submit**.

### 8.8.6 Recovering from a system crash



**Figure 8-1: System crash decision tree**

1. See “Using ftintegrity” on page 94.
2. See “Scripted federation restore” on page 230.
3. See “Deleting a collection and recreating indexes” on page 207.

# Chapter 9

## Automated Utilities (CLI)

### 9.1 CLI properties and environment

The CLI tool is located in `xplore_home/dsearch/admin`. The tool wrapper is `xplore.bat` (Windows) or `xplore.sh` (Linux). Edit the properties file `xplore.properties` to set the environment for the CLI execution.

**Table 9-1: CLI properties**

Property	Description
host	Primary xPlore instance host: fully qualified hostname or IP address
port	Primary xPlore instance port. If you change to the HTTPS protocol, change this port.
password	xPlore administrator password, set up when you installed xPlore (same as xPlore administrator login password)
verbose	Prints all admin API calls to console. Default: true. For batch scripts, set to <code>false</code> .
protocol	Valid values: <code>http</code> or <code>https</code>

The CLI uses the Groovy script engine.

### 9.2 Using the CLI

1. Open a command-line window.
2. Change the working directory to the location of `xplore_home/dsearch/admin`.
3. Run a CLI command with no parameters, use the following syntax appropriate for your environment (Windows or Linux). The command is case-insensitive.

```
xplore.bat <command>
./xplore.sh <command>
```

Examples:

```
xplore.bat resumeDiskWrites
./xplore.sh resumeDiskWrites
```

4. Run a CLI command with parameters using the following syntax appropriate for your environment (Windows or Linux). The command is case-insensitive. Use double quotes around the command, single quotes around parameters, and a forward slash for all paths:

```
xplore.bat "<command> [parameters]"  
./xplore.sh "<command> [parameters]"
```

Examples:

```
xplore "backupFederation 'c:/xPlore/dsearch/backup', true, null"  
./xplore.sh "dropIndex 'dftxml', 'folder-list-index' "
```

The command executes, prints a success or error message, and then exits.

5. (Optional) Run CLI commands from a file using the following syntax:

```
xplore.bat -f <filename>
```

Examples:

```
xplore.bat -f file.txt  
./xplore.sh -f file.txt
```

Example:

Call the wrapper without a parameter to view a help message that lists all CLIs and their arguments. For example:

```
xplore help backupFederation  
./xplore.sh help backupFederation
```

### 9.3 CLI batch file

Use the following syntax to reference a batch file containing xPlore commands. Substitute the full path to your batch file:

```
xplore.bat -f <batch_file_name>  
or  
xplore.sh -f <batch_file_name>
```

For example, the following batch file *sample.gvy* suspends index writes and performs a federation backup. Use a forward slash for paths.

```
suspendDiskWrites  
folder='c:/folder'  
isIncremental=false  
backupFederation folder, isIncremental, null  
println 'Done'
```

Call the batch file with a command like the following:

```
xplore.bat -f sample.gvy
```

## 9.4 Scripted backup

The default backup location is specified in `indexserverconfig.xml` as the value of the `path` attribute on the element `admin-config/backup-location`. Specify any path as the value of `[backup_path]`.

### Scripted federation backup

```
xplore backupFederation [backup_path], <is_incremental>, null
```

`is_incremental`

Boolean. Set to false for full backup, true for incremental. For incremental backups, set `keep-xdb-transactional-log` to true in xPlore administrator. Choose **Home > Global Configuration > Engine**.

Examples:

```
xplore "backupFederation null, false, null"
xplore "backupFederation 'c:/xplore/backup', false, null"
```

### Scripted domain backup

```
xplore backupDomain <domain_name>, [backup_path]
```

Examples:

```
xplore "backupDomain 'myDomain', null"
xplore "backupDomain 'myDomain', 'c:/xplore/backup' "
```

### Scripted collection backup

```
xplore backupCollection collection(<domain_name>, <collection_name>), [backup_path]
```

Examples:

```
"backupCollection collection('myDomain', 'default'), null"
"backupCollection collection('myDomain', 'default'), 'c:/xplore/backup' "
```

### Scripted file or volume backup

Stop the index agent and suspend disk writes before backup. Resume disk writes and the index agent after backup.

```
"suspendDiskWrites"
"resumeDiskWrites"
```

## 9.5 Scripted federation restore

Back up and restore your jars or DLLs for custom content processing plugins or annotators.

[backup\_path]: Path to your backup file. If not specified, the default backup location in indexserverconfig.xml is used: The value of the *path* attribute on the element *admin-config/backup-location*. Specify any path as the value of .

1. Stop all xPlore instances.
2. Delete existing data files:
  - All files under *xplore\_home*/data
  - All files under *xplore\_home*/config.
3. Run the restore CLI. If no path is specified, the default location in indexserverconfig.xml is used.

```
"restoreFederation '[backup_path]' "
```

For example:

```
xplore "restoreFederation 'C:/xPlore/dsearch/backup/federation/2011-03-23-16-02-02'"
```

4. Restart all xPlore instances.



**Note:** If you are restoring a federation and a collection that was added after the federation backup, do the following:

1. Restore the federation.
2. Start up and shut down all xPlore instances.
3. Restore the collection.

## 9.6 Scripted domain restore

[backup\_path]: Path to your backup file. If not specified, the default backup location in indexserverconfig.xml is used: The value of the *path* attribute on the element *admin-config/backup-location*. Specify any path as the value of .

[bootstrap\_path]: Path to the bootstrap file in the WEB-INF classes directory of the xPlore primary instance.

1. Force-detach the domain using the CLI detachDomain. The second argument is whether to force detach. Force detach can corrupt the data. Use only for restore operations.

```
"detachDomain '[domain_name]', true"
```

For example:

```
xplore "detachDomain 'defaultDomain', true"
```

2. Generate the orphaned segment list using the CLI `listOrphanedSegments`. If an orphaned segment file is not specified, the IDs or orphaned segments are sent to `stdio`. See “[Orphaned segments CLIs](#)” on page 233.
3. Stop all xPlore instances.
4. Run the restore CLI. If no bootstrap path is specified, the default location in the WEB-INF classes directory of the xPlore primary instance is used.
 

```
"restoreDomain '[backup_path]', '[bootstrap_path]' "
```
5. Restart xPlore instances.
6. If orphaned segments are reported before restore, run the CLI `purgeOrphanedSegments`. If an orphaned segment file is not specified, the segment IDs are read in from `stdin`. See “[Orphaned segments CLIs](#)” on page 233.
7. Force-attach the domain using the CLI `attachDomain`.
 

```
xplore "attachDomain '[domain_name]', true"
```

For example:

```
xplore "attachDomain 'defaultDomain', true"
```
8. Perform a consistency check and test search.
9. Run the ACL and group replication script `aclreplication_for_repository_name` in `xplore_home/setup/indexagent/tools` to update any security changes since the backup. See “[Manually updating security](#)” on page 64.
10. Run `ftintegrity`. See “[Using ftintegrity](#)” on page 94.

## 9.7 Scripted collection restore

`[backup_path]`: Path to your backup file. If not specified, the default backup location in `indexserverconfig.xml` is used: The value of the `path` attribute on the element `admin-config/backup-location`. Specify any path.

`[bootstrap_path]`: Path to the bootstrap file in the WEB-INF classes directory of the xPlore primary instance.

1. Set the collection state to off-line.
2. Force-detach the collection using the CLI `detachCollection`. The last argument specifies a forced detachment. Use only for restore, because force-detach can corrupt the collection.
 

```
xplore"detachCollection collection('[domain_name>', '[collection_name]'), true"
```
3. Generate the orphaned segment list using the CLI `listOrphanedSegments`. If an orphaned segment file is not specified, the IDs or orphaned segments are sent to `stdio`. See “[Orphaned segments CLIs](#)” on page 233.
4. Stop all xPlore instances.

5. Run the restore CLI. If no bootstrap path is specified, the default location in the WEB-INF classes directory of the xPlore primary instance is used.

```
"restoreCollection, [backup_path], [bootstrap_path]"
```

6. Restart xPlore instances.
7. If orphaned segments are reported before restore, run the CLI `purgeOrphanedSegments`. If an orphaned segment file is not specified, the segment IDs are read in from `stdin`.
8. Force-attach the collection using the CLI `attachCollection`. The last argument is for forced attachment.

```
xplore "attachCollection '[domain_name]', '[collection_name]', true"
```

## 9.8 Force detach and attach CLIs

You detach a domain or collection before you restore it. You attach it after you restore it.

1. Detach syntax:

```
"detachDomain '[domain_name]', 'true' "
or
"detachCollection collection('[domain_name]', '[collection_name]'), true"
```

Examples:

```
"detachDomain ''myDomain', true"
or
"detachCollection collection('myDomain','default'), true"
```

2. Attach syntax:

```
"attachDomain ''[domain_name]', true"
or
"attachCollection collection('[domain_name]', '[collection_name]'), true"
```

Examples:

```
"attachDomain 'myDomain', true"
or
"attachCollection collection('myDomain','default'), true"
```

## 9.9 Orphaned segments CLIs

Segments can be orphaned when content is added to or removed from the index after backup. The restore operation does not reflect these changes, and any new segments that were used after backup are orphaned. The xDB database in xPlore does not start up with orphaned segments unless you force a restart. See “[Troubleshooting data management](#)” on page 209. Federation restore does not create orphaned segments.

1. List orphaned segments before restore. If [orphan\_file\_path] is not specific, the IDs of orphaned segments are sent to *stdio*. This file is used to purge orphaned segments after restore. Syntax:

```
"listOrphanedSegments collection|domain [backup_path]
[orphan_file_path] [bootstrap_path]"
```

For example:

```
"listOrphanedSegments 'domain', 'backup/myDomain/2012-10',
c:/temp/orphans.lst'
'C:/xplore/wildfly9.0.1/server/DctmServer_PrimaryDsearch/deployments/dsearch.war/
WEB-INF/classes/indexserver-bootstrap.properties' "
or
"listOrphanedSegments 'collection', 'backup/myDomain/default/2012-10', null,
'C:/xplore/wildfly9.0.1/server/DctmServer_PrimaryDsearch/deployments/dsearch.war/
WEB-INF/classes/indexserver-bootstrap.properties' "
```

2. If orphaned segments are reported before restore, run the CLI `purgeOrphanedSegments`. If [orphan\_file\_path] is not specified, the segment IDs are read in from *stdin*. For file path, use forward slashes. Syntax:

```
xplore purgeOrphanedSegments '[orphan_file_path]'
```

For example:

```
purgeOrphanedSegments 'c:/temp/orphans.lst'
or
purgeOrphanedSegments null
```

*Arguments:*

- [backup\_path]: Path to your backup file. If not specified, the default backup location in `indexserverconfig.xml` is used: The value of the `path` attribute on the element `admin-config/backup-location`. Specify any path as the value of .
- [bootstrap\_path]: Path to the bootstrap file in the WEB-INF classes directory of the xPlore primary instance.

## 9.10 Domain mode CLIs

If a domain index is corrupt, use the CLI setDomainMode to set the mode to maintenance. In maintenance mode, the only allowed operations are repair and consistency check. Queries are allowed only from xPlore administrator.

The mode does not persist: it reverts to *normal* on xPlore restart.

Syntax:

```
"setDomainMode [domain_name], [maintenance|normal]"
```

For example:

```
"setDomainMode 'myDomain', 'maintenance' "
```

## 9.11 Collection and domain state CLIs

Syntax:

```
"setState domain|collection, [domain_name],[collection_name],[state]"
```

*Domain state*

Set collection\_name to null. Valid states: read\_only and reset.

Example:

```
"setState 'domain', 'myDomain', null, 'reset' "
```

*Collection state*

Valid states: index\_and\_search (read/write), update\_and\_search (read and update existing documents only), search\_only, index\_only (write only), and off\_line. The update\_and\_search state changes a flag so that new documents cannot be added to the collection. Existing documents can be updated. The state change is not propagated to subcollections.

Example:

```
"setState 'collection', 'myDomain', 'default', 'off_line' "
```

## 9.12 Activate spare instance CLI

You can install a spare instance that you activate when another instance fails. For instructions on activating the spare using xPlore administrator, see “[Replacing a failed instance with a spare](#)” on page 40.

 **Note:** This CLI can be used only to replace a secondary instance. To replace a primary instance, see “[Replacing a failed primary instance](#)” on page 41.

Syntax:

```
"activateSpareNode '[failed_instance]', '[spare_instance]' "
```

Example:

```
"activateSpareNode 'node2', 'spare1' "
```

## 9.13 Detecting the version of an instance

Use the CLI showVersion to get the version of the instance on the current host.

```
xplore.bat showVersion  
./xplore.sh showVersion
```

## 9.14 Cleaning up after failed index rebuild

When an index rebuild fails, the collection configuration in indexserverconfig.xml is annotated with a property named “Build\_dmftdoc”. A temporary index folder is left behind. You may see the following error message in dsearch.log:

```
Xhive exception message: INTERRUPTED
```

To remove the rebuild index property and clean up the temporary index folder, use the following CLI:

Syntax:

```
removeInConstructionIndexes [domain], [collection]
```

For example:

```
xplore "removeInConstructionIndexes 'myDocbase',  
'myCollection' "  
./xplore.sh "removeInConstructionIndexes 'myDocbase',  
'myCollection' "
```

## 9.15 Final merge CLIs

You can use CLI commands to manually start and stop merges and view merge status.

Final merge is recorded in the audit record as a FINAL\_MERGE event.

### Getting merge status

Use the CLIs isFinalMergeOngoing and getFinalMergeStatus.

Syntax:

```
isFinalMergeOngoing [domain], [collection]
getFinalMergeStatus [domain], [collection]
```

For example:

```
xplore "isFinalMergeOngoing 'myDocbase', 'myCollection' "
./xplore.sh "getFinalMergeStatus 'myDocbase', 'myCollection' "
```

You can also see merge status using xPlore administrator. A **Merging** icon is displayed during the merge progress.

### Starting and stopping final merge

Use the CLIs startFinalMerge and stopFinalMerge.

Syntax:

```
startFinalMerge [domain], [collection]
stopFinalMerge [domain], [collection]
```

For example:

```
xplore "startFinalMerge 'myDocbase', 'myCollection' "
./xplore.sh "stopFinalMerge 'myDocbase', 'myCollection' "
```

## 9.16 Collecting diagnostic information

To collect all logs, config files, system data, and session information, do the following:

1. Open CLI, and cd into %XPORE%/dsearch/admin.

2. Run the following command:

```
xplore.bat/sh -f scripts/diagnostic.groovy -h
```

3. Do one of the following:

- If the primary instance is online, run this command to export all information:

```
xplore.bat/sh -f scripts/diagnostic.groovy
```

- If the primary instance is offline, run this command to ignore runtime session information and audit/metrics DB:

```
xplore.bat/sh -f scripts/diagnostic.groovy -isi -iam
```

4. Use the <-file <file\_name>> option to specify a file that includes a list of folders for export.

The default target location is . /dump. To change this, use the <-dest <dest\_folder>> option.



# Chapter 10

## Search

### 10.1 About searching

Specific attributes of the dm\_sysobject support full-text indexing. Use Documentum Administrator to make object types and attributes searchable or not searchable and to set allowed search operators and default search operator.

- Set the *is\_searchable* attribute on an object type to allow or prevent searches for objects of that type and its subtypes. Valid values: 0 (false) and 1 (true). The client application must read this attribute. (The indexing process does not use it.) If *is\_searchable* is false for a type or attributes, Webtop does not display them in the search UI. Default: true.
- Set *allowed\_search\_ops* to set the allowed search operators and *default\_search\_op* to set the default operator. Valid values for *allowed\_search\_ops* and *default\_search\_op*:

Value	Operator
1	=
2	<>
3	>
4	<
5	>=
6	<=
7	begins with
8	contains
9	does not contain
10	ends with
11	in
12	not in
13	between
14	is null
15	is not null
16	not

- The *default\_search\_arg* attribute sets a default argument for the default operator. The client must read these attributes; the indexing process does not use them. Webtop displays the allowed operators and the default operator.

Documentum Server client applications issue queries through the DFC search service or through DQL. DFC 6.6 and higher translates queries directly to XQuery for xPlore. DQL queries are handled by the Documentum Server query plugin, which translates DQL into XQuery unless XQuery generation is turned off. Not all DQL operators are available through the DFC search service. In some cases, a DQL search of the Server database returns different results than a DFC/xPlore search. For more information on DQL and DFC search differences, see “[DQL, DFC, and DFS queries](#)” on page 276.

DFC generates XQuery expressions by default. If XQuery is turned off in DFC, FTDQL queries are generated. The FTDQL queries are evaluated in the xPlore server. If all or part of the query does not conform to FTDQL, that portion of the query is converted to DQL and evaluated in the Documentum Server database. Results from the XQuery are combined with database results. For more information on FTDQL and SDC criteria, see the *Documentum Server DQL Reference Guide*.

xPlore search is case-insensitive and ignores white space or other special characters. Special characters are configurable.

### 10.1.1 Query operators

Operators in XQuery expressions, DFC, and DQL are interpreted in the following ways:

- DQL operators: All string attributes are searched with the *ftcontains* operator in XQuery. All other attribute types use value operators (= != <>). In DQL, dates are automatically normalized to UTC representation when translated to XQuery.
- DFC: When you use the DFC interface *IDfXQuery*, your application must specify dates in UTC to match the format in *dftxml*.
- XQuery operators
  - The value operators = != <> specify a value comparison search. Search terms are not tokenized. Can be used for exact match or range searching on dates and IDs.

Any subpath that can be searched with a value operator must have the *value-comparison* attribute set to *true* for the corresponding subpath configuration in *indexserverconfig.xml*. For example, an improper configuration of the *r\_modify\_date* attribute sets *full-text-search* to *true*. A date of ‘2010-04-01T06:55:29’ is tokenized into 5 tokens: ‘2010’ ‘04’ ‘01T06’ ‘55’ ‘29’. A search for ‘04’ returns any document modified in April. The user gets many non-relevant results. Therefore, *r\_modify\_date* must have *value-comparison* set to *true*. Then the date attribute is indexed as one token. A search for ‘04’ would not hit all documents modified in April.

- The *ftcontains* operator (XQFT syntax) specifies that the search term is tokenized before searching against index.

If a subpath can be searched by *ftcontains*, set the *full-text-search* attribute to *true* in the corresponding subpath configuration in *indexserverconfig.xml*.

## 10.2 Administering search

### Common search service tasks

You can configure all search service parameters by choosing **Global Configuration** from the **System Overview** panel in xPlore administrator. The default values have been optimized for most environments.

#### *Enabling search*

Enable or disable search by choosing an instance of the search service in the left pane of the administrator. Click **Disable** (or **Enable**).

#### *Cancelling running queries*

Open an instance and choose **Search Service**. Click **Operations**. All running queries are displayed. Click a query and delete it.

#### *Viewing search statistics*

Choose **Search Service** and click an instance:

- Accumulated number of executed queries
- Number of failed queries
- Number of pending queries
- Number of spooled queries
- Number of execution plans
- Number of streamed results
- Maximum query result batch size
- Total result bytes
- Maximum hits returned by a query
- Average number of results per query request
- Maximum query execution time
- Average duration of query execution
- Queries per second

Use reports for additional information about queries. Search auditing must be turned on to accumulate the report data. (It is on by default.) See “[Search reports](#)” on page 350.

- “[Configuring query warmup](#)” on page 242
- “[Configuring scoring and freshness](#)” on page 246
- “[Adding a thesaurus](#)” on page 252

- “Configuring query summaries” on page 259
- “Configuring query lemmatization” on page 257

### 10.2.1 Configuring query warmup

You can warm up the system at xPlore startup. Queries are logged in the audit record for each xPlore instance. The warmup utility on each instance warms up queries that were run on that instance.

Queries are warmed up from the audit log or from a file (file first, then user queries.) Warmup is typically done through the audit log, and security is evaluated for these queries by default. The most recent queries from the last N users (N stands for the value of *number\_of\_unique\_users* in query.properties) are played.

In warmups for file-based queries, security is not evaluated by default. If you need security evaluation, set the following properties in query.properties: *security\_eval*, *user\_name*, *super\_user*.

All types of warmups follow the same configurable schedule and their activities are recorded in the same log file.

#### Enabling or disabling warmup

Edit indexserverconfig.xml. For information on editing this file, see [“Modifying indexserverconfig.xml” on page 54](#). Locate the performance element. If it does not exist, add it with the following content:

```
<performance>
<warmup status="on">
<properties>
<property value=".../dsearch/xhive/admin/QueryRunner.bat"
name="script"/>
<property value="600" name="timeout-in-secs"/>
</properties>
</warmup>
</performance>
```

Set the child element *warmup* status to on or off. You can set the warmup timeout in seconds. If the warmup hangs, it is canceled after this timeout period.



**Note:** By default, the relative path to the warmup script QueryRunner.bat (Windows) or QueryRunner.sh (Linux) in indexserverconfig.xml assumes that the configuration directory is *xplore\_home/config*. If not, you must modify the path to a corresponding relative path or an absolute path for the warmup script to run correctly.

#### Configuring warmup

Configure warmup in query.properties. This file is in *xplore\_home/dsearch/xhive/admin*. Restart all xPlore instances for your changes to take effect.

**Table 10-1: Auto-warmup configuration**

<b>Key</b>	<b>Description</b>
xplore_qrserver_host	Primary xPlore instance host. Not needed for file-based warmup.
xplore_qrserver_port	Port for primary xPlore instance. Not needed for file-based warmup.
xplore_protocol	Communication protocol used for file-based warmup. Valid values: <i>http</i> and <i>https</i>
xplore_domain	Name of domain in xPlore (usually the same as the Documentum Server name). You must change this to a valid domain. This is used only for file-based warmup. An incorrect domain name is recorded in <i>Querywarmup.log</i> as <i>FtSearchException: Invalid domain</i> .
security_eval	Evaluate security for queries in warmup. Default: false. Used only for file-based warmup.
user_name	Name of user or superuser who has permission to run warmup queries. Required if <i>security_eval</i> is set to true. Used only for file-based warmup.
super_user	Set to true if the user who runs warmup queries is a Documentum Server superuser. Required if <i>security_eval</i> is set to true.
query_file	Specify a file name that contains warmup queries. Query can be multi-line, but the file cannot contain empty lines. If no name is specified, queries are read from the audit record, (Query auditing is enabled by default.)
batch_size	Set number of results in one batch for warmup queries. Default: 20.
timeout	Set maximum milliseconds to try a warmup query. Default: 60000.
max_retries	Set maximum number of times to retry a failed query. Default: 10.
print_result	Set to true to print results to <i>queryWarmer.log</i> in <i>xplore_home/dsearch/xhive/admin/logs</i> . Default: false.
fetch_result_byte	Maximum number of bytes to fetch in a warmup query. Default: 4096000.

Key	Description
read_from_audit	Set to true (default) to read queries from audit record. Requires query auditing enabled (default). (See <a href="#">"Auditing queries" on page 302</a> .) Warmup is read first from a file if query_file is specified. Default: true.
number_of_unique_users	Number of users in audit log for whom to replay queries. Default: 10
number_of_queries_per_user	Number of queries for each user to replay. Default: 1
schedule_warmup	Enables the warmup schedule. Default: true.
schedule_warmup_period	How often the warmup occurs. For example, a value of 1 in DAYS units results in daily warmup. Default: 1.
schedule_warmup_units	Valid values: DAYS   HOURS   MINUTES   SECONDS   MILLISECONDS   MICROSECONDS. Default: DAYS.
initial_delay	Start warmup after the specified initial delay, and subsequently after schedule_warmup_period. Default: 0. If any execution of the task encounters an exception, subsequent executions are suppressed.
query_response_time	Select the queries from audit records for replay that has query response time (fetch time + execution time) in milliseconds lower or equal this value. Default: 60000.
exclude_users	Select the queries from audit records that are not run by these users. Set a comma-separated list of users. Default: unknown.
check_node_status_interval	How much time in milliseconds warmup waits before it checks again the status of nodes when one or more nodes are not successfully started in a multi-node environment. Warmup runs only after all the nodes are started. Default: 60000.
check_node_status_max_retries	Number of times to check the status of nodes when one or more nodes are not successfully started in a multi-node environment. Warmup runs only after all the nodes are started. Set it to -1 to check indefinitely. Default: 15.

## Auto-skipping scheduled warmup under heavy load

When warmup is configured to be a scheduled activity, it might add unexpected load to the system, especially when there are many concurrent queries or GC

overhead is already high. To avoid affecting normal queries, the system automatically skips a scheduled warmup under heavy load. More precisely, the system skips the current scheduled warmup when the one of the following conditions is true:

- Index requests are throttled.
- Search requests are throttled.
- GC overhead exceeds the limit configured in the *throttle-gc-busyness-threshold* property under *node->properties* in *indexserverconfig.xml*, which defaults to 0.4.

## Sample query file (queries.xq)

Queries in the query file can be multi-line, but the file cannot contain empty lines. The empty sample queries.xq is in *xplore\_home/dsearch/xhive/admin*.

```
declare option xhive:fts-analyzer-class 'com.emc.documentum.core.fulltext.indexserver.core.index.xhive.IndexServerAnalyzer';
declare option xhive:ignore-empty-fulltext-clauses 'true';
declare option xhive:index-paths-values 'dmftmetadata//owner_name,dmftsecurity/acl_name,
dmftsecurity/acl_domain,/dmftinternal/r_object_id';
for $i score $ in collection('/dm_notes/dsearch/Data') /dmftdoc[ ( ( dmftinternal/i_all_types = '030000018000010d' ) and ( ( dmftversions/iscurrent = 'true' ) ) and ( (. ftcontains ( (( 'augmenting') with stemming)) using stop words ("") ) ) )
order by $ descending
return <dmrow>{if ($i/dmftinternal/r_object_id
then $i/dmftinternal/r_object_id
else <r_object_id/>){if ($i/dmftsecurity/ispublic)
then $i/dmftsecurity/ispublic
else <ispublic/>}{if ($i/dmftinternal/r_object_type)
then $i/dmftinternal/r_object_type
else <r_object_type/>}{if ($i/dmftmetadata/*/owner_name)
then $i/dmftmetadata/*/owner_name else <owner_name/>{
if ($i/dmftvstamp/i_vstamp)
then $i/dmftvstamp/i_vstamp else <i_vstamp/>}{xhive:highlight(
$i/dmftcontents/dmftcontent/dmftcontentref)}</dmrow>
```

## Warmup logging

Warmup activity is logged in the audit record and reported in the admin report Audit Records for Search Component:

- Warmup is logged in the file *queryWarmer.log*, located in *xplore\_home/dsearch/xhive/admin/logs*. Use this log to verify when a collection was last warmed up.
- All the queries that are replayed for warmup from a file or the audit record are tagged as a QUERY\_WARMUP event in the audit records. The log includes the query to get warmup queries. You can see this type in the admin report **Top N Slowest Queries**.

To view all warmup queries in the audit record, run the report **Audit records for search component** (query type: Warmup Tool Query (QUERY\_WARMUP)) in xPlore administrator.

## 10.2.2 Configuring scoring and freshness

xPlore uses the following ranking principles to score query results:

- If a search term appears more than once in a document, the hit is ranked higher than a document in which the term occurs only once. This comparison is valid for documents of the same content length.
- If a search term appears in the metadata, the hit is ranked higher than when the term occurs in the content.
- When the search criteria are linked with OR, a document with both terms is ranked higher than a document with one term.
- If the search spans multiple instances, the results are merged based on score.

Other scoring:

- For Documentum DFC clients that have a ranking configuration file, the ranking file defines how the source score (xPlore) and the DFC score are merged. The xPlore score could be ignored, partially used, or fully used without any other influence.
- The DFC client application can specify a set of Documentum attributes for sorting results using the API in an IDfQueryBuilder API. If the query contains an order by attribute, results are returned based on the attribute and not on the computed score.

These ranking principles are applied in a complicated Lucene algorithm. The Lucene scoring details are logged when xDB logging is set to TRACE. See “[Configuring logging](#)” on page 357.

Some of the following settings require reindexing. Freshness is supported by default.

1. Edit indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.
2. Add a boost-value attribute to a *sub-path* element. The default boost-value is 1.0. A change requires reindexing. In the following example, a hit in the keywords metadata increases the score for a result:

```
<sub-path returning-contents="true" boost-value="2.0" path="dmftmetadata/keywords"/>
```

3. By default the Documentum attribute *r\_modify\_date* is used to boost scores in results (freshness boost). You can remove the freshness boost factor, change how much effect it has, or boost a custom date attribute.
  - To remove this boost, edit indexserverconfig.xml and set the property *enable-freshness-score* to *false* on the parent *category* element. This change affects only query results and does not require reindexing.

```
<category name='dftxml'><properties>
  ...
</properties>
</category>
```

```
<property name="enable-freshness-score" value="false" />
</properties></category>
```

- Change the freshness boost factor. Changes do not require reindexing. Only those documents for which you set a boost for a custom date between six years old and 75 days in future have a freshness factor. The weight for freshness is equal to the weight for the Lucene relevancy score. Set the value of the property *freshness-weight* in index-config/properties to a decimal between 0 (no boost) and 1.0 (override the Lucene relevancy score). For example:

```
<index-config><properties>
...
<property name="enable-subcollection-ftindex" value="false"/>
<property name="freshness-weight" value="0.75" />
```

- To boost a different date attribute, specify the path to the attribute in dftxml as the value of a freshness-path property. This change requires reindexing. In the following example, the r\_creation\_date attribute is boosted:

```
<index-config><properties>
...
<property name="enable-subcollection-ftindex" value="false"/>
<property name="freshness-weight" value="0.75" />
<property name="freshness-path" value="dmftmetadata/.*/r_creation_date" />
```

4. Configure weighting for query term source: original term. This does not require reindexing. Edit the following properties in search-config/properties. The value ranges from 1 to 1000.

```
<property name="query-original-term-weight" value="100.0" />
```

5. Restart the xPlore instances.

### 10.2.3 Supporting search in XML documents

When a document to be indexed contains XML content, you can specify how the XML markup and content should be handled:

- Whether XML content can be tokenized (*tokenize="true | false"*).
- Whether XML content can be stored within the input document (*store="embed | none"*).



**Note:** When the *store* attribute is set to *none*, XML documents are not tokenized even if the *tokenize* attribute is set to *true*.

- Whether XML attributes can be indexed and searched (*include-attribute-value="true | false"*). See “[XML attribute support](#)” on page 180.

Set these attributes on the for-element-with-name/xml-content element in indexserverconfig.xml.

All XML documents are parsed and entities are expanded unless the document size exceeds max\_text\_threshold. CPS stores tokens for documents that exceed this size, but they are not embedded in the index and summaries cannot be calculated.

By default, the XML structure (element names and attributes of an input document) is not indexed. Only the values of XML elements are extracted and indexed.

When the XML content size is less than the threshold (*file-limit* on the `xml-content` element), xPlore parses the XML content and extracts the text. The extracted text is added to the `dmftcontentref` element in `dmftxml`.

You can configure xPlore to support XML zone search. XML structure is preserved along with the content of the elements. The elements, attributes, and content can be searched.

DTD URLs are resolved. External entities are expanded and added to the XML content. DTDs and external entities can be placed on the xPlore host.



**Note:** When the size of the file exceeds the size that is specified as *file-limit* on the `xml-content` element in `indexserverconfig.xml`, XML element values are embedded into the `dmftcontentref` element of the `dmftxml` record. You can still do a full-text search on the file contents if it does not fail XML parsing.

## Adding DTD and external entity support

External DTDs with a URL are resolved by the CPS parser. To successfully resolve external DTDs and entities, create a directory on the CPS host and save these files in that directory. You must also edit the CPS configuration file, such as `PrimaryDsearch_local_configuration.xml` and add the `xml-entity-file-path` property with the value set to this directory. To apply the changes, restart the CPS instance.

The following XML snippet provides an example of a CPS configuration file:

```
<cps_config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://cps.cma.emc.com/configuration.xsd"      xmlns="http://
cps.cma.emc.com/"> ...      <property value="C:\DTDFiles" name="xml-entity-file-path"/>
<cps_pipeline>...
```



### Notes

- The property `xml-entity-file-path` must be located before `<cps_pipeline>`.
- If your documents containing XML have already been indexed, they must be reindexed to include parsing with the DTD and entities.

## Disabling DTD Validation

You can disable XML DTD validation by editing the CPS configuration file, such as `PrimaryDsearch_local_configuration.xml` to add the `xml-disable-dtd` property with the value `true`, and then restart the CPS instances.

The following XML snippet provides an example of disabling DTD validation:

```
<cps_config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://cps.cma.emc.com/configuration.xsd"      xmlns="http://
```

```
cps.cma.emc.com/"> ... <property name="xml-disable-dtd" value="true" />
<cps_pipeline>
```

### Notes

- The property `xml-disable-dtd` must be located before `<cps_pipeline>`.
- If your documents containing XML have already been indexed, you must reindex them to parse them with DTD validation disabled.

## Adding XML zone search support

To support searching on XML content or attribute values (also known as zone search):

1. Change `index-as-sub-path` to true on the `xml-content` element in `indexserverconfig.xml`. XML elements and their content are embedded into `dmftxml` as children of the `dmftcontentref` element. Attributes on the elements are preserved.

When `index-as-sub-path` is set to true, the language identification relies on the metadata elements configured in `element-for-language-identification` on the `linguistic-process` element. If the language of content is different from the language of metadata, the content is not indexed correctly. Incorrect tokens are stored and a query against content can fail to return expected results.

2. Change the following sub-path configurations:

```
<sub-path path="dmftcontents/dmftcontent///*" full-text-search="false" />
```

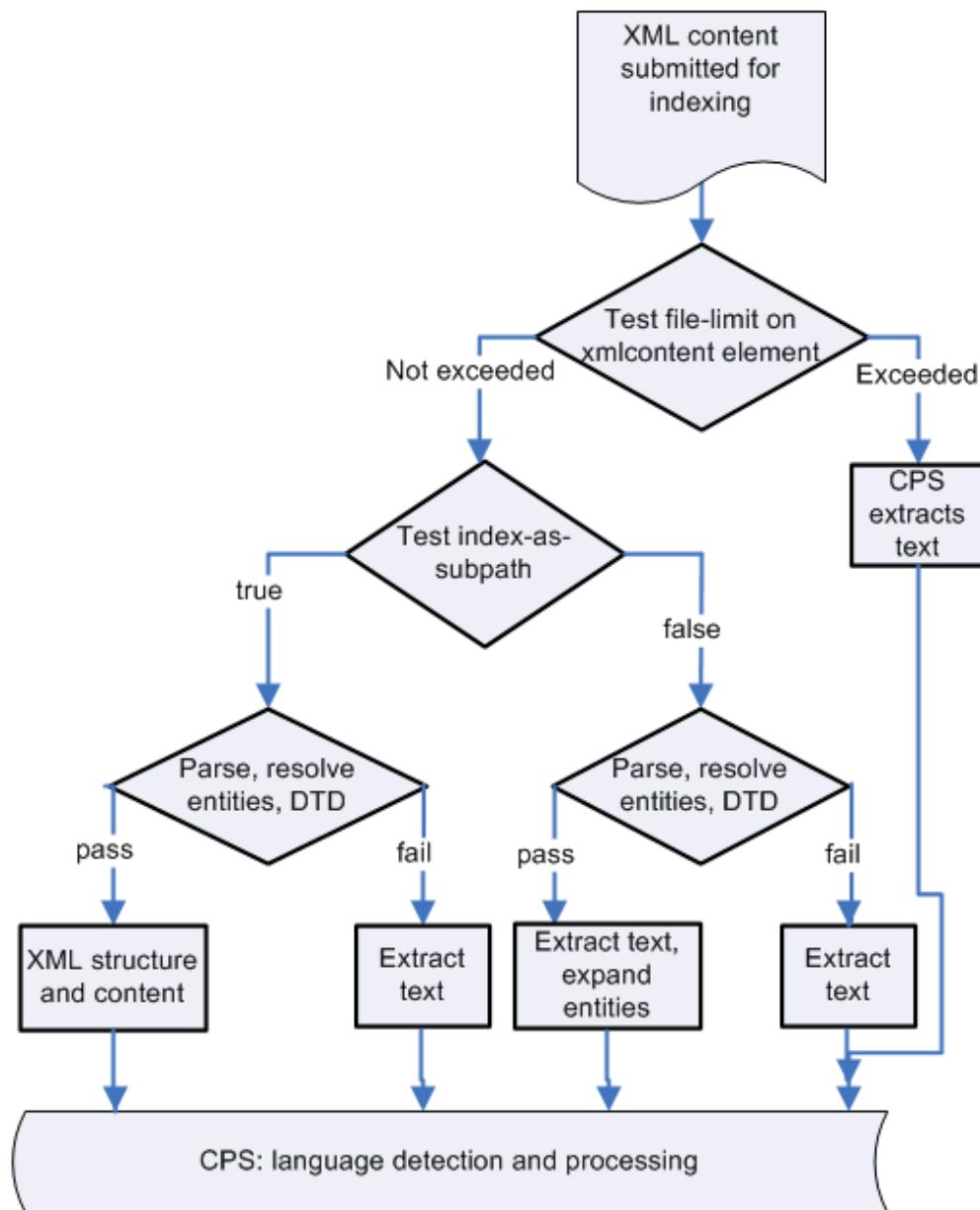
Remove `full-text-search="false"` in the definition:

```
<sub-path path="dmftcontents/dmftcontent///*" />
```

### Notes

- If your documents containing XML have already been indexed, they must be reindexed to include the XML content.
- If the content exceeds the CPS `file-limit` in `xml-content`, XML content is not embedded.

The following illustration shows how XML content is processed depending on your configuration. The table assumes that the document submitted for indexing does not exceed the size limit in index agent configuration and the content limit in CPS configuration.

**Figure 10-1: XML processing options**

### Handling XML parsing errors

Specify how to handle parsing errors with the on-embed-error attribute. Set this attribute on the for-element-with-name/xml-content element. Valid values: embed\_as\_cdata | ignore | fail. The option embed\_as\_cdata stores the entire XML content as a CData child element of the specified element, for example, dmftcontentref. The ignore option does not store the XML markup. For the fail option, none of the content in the document is searchable.

## Zone search in XQueries

DFC client applications like Webtop or D2 generate XQueries, but they cannot perform zone searches. In the following example, an XML document has the following structure:

```
<company>
  <staff>
    <firstname>Michelle</firstname>
    <lastname>Specht</lastname>
    <nickname>mspecht</nickname>
    <salary>100000</salary>
  </staff>
  <staff>
    <firstname>John</firstname>
    <lastname>Heidrich</lastname>
    <nickname>jheidrich</nickname>
    <salary>200000</salary>
  </staff>
  ...1000 more entries
</company>
```

A search for the word *staff* generates the following simple XQuery:

```
let $j:= for $x in collection('/XMLTest')/dmftdoc
[. ftcontains 'staff' with stemming]
```

You can use IDfXQuery to generate the following query, which is much more specific and performs better:

```
let $j = for $x in collection('/XMLTest')/dmftdoc
[dmftcontents/dmftcontent/dmftcontentref/company/staff
 ftcontains 'John' with stemming]
```

## Rewriting VQL queries

Older Documentum applications supported XML zone searches with Verity Query Language (VQL). FAST indexing did not support VQL queries. With xPlore, you can configure zone search support using index-as-sub-path, or you can rewrite some VQL queries to XQuery equivalents.

- Perform structured searches of XML documents using XQuery or the DFC interface IDfXQuery.
- Join different objects using DQL (NOFTDQL), XQuery, or the DFC interface IDfXQuery.
- Denormalize the relationship of a document to other objects or tables, such as email attachments, using XQuery or the DFC interface IDfXQuery.
- Perform boolean searches using DQL, XQuery, or the DFC interface IDfXQuery.

For a table of VQL examples and their equivalents in XQuery expression, see “[VQL and XQuery Syntax Equivalents](#)” on page 420.

## 10.2.4 Adding a thesaurus

A thesaurus provides results with terms that are related to the search terms. For example, when a user searches for *car*, a thesaurus expands the search to documents containing *auto* or *vehicle*. When you provide a thesaurus, xPlore expands search terms in full-text expressions to similar terms. This expansion takes place before the query is tokenized. Terms from the query and thesaurus expansion are highlighted in search results summaries.

A thesaurus can have terms in multiple languages. Linguistic analysis of all the terms that are returned, regardless of language, is based on the query locale.

The thesaurus must be in SKOS format, a W3C specification. FAST-based thesaurus dictionaries must be converted to the SKOS format. Import your thesaurus to the file system on the primary instance host using xPlore administrator. You can also provide a non-SKOS thesaurus by implementing a custom class that defines thesaurus expansion behavior. See [“Adding custom access to a thesaurus” on page 328](#).

### SKOS format

The format starts with a concept (term) that includes a preferred label and a set of alternative labels. The alternative labels expand the term (the related terms or synonyms). Here is an example of such an entry in SKOS:

```
<skos:Concept rdf:about="http://www.my.com/#canals">
<skos:prefLabel>canals</skos:prefLabel>
<skos:altLabel>canal bends</skos:altLabel>
<skos:altLabel>canalized streams</skos:altLabel>
<skos:altLabel>ditch mouths</skos:altLabel>
<skos:altLabel>ditches</skos:altLabel>
<skos:altLabel>drainage canals</skos:altLabel>
<skos:altLabel>drainage ditches</skos:altLabel>
</skos:Concept>
```

xPlore supports SKOS two-way expansion, so when searching for a term defined by a preferred label or alternative labels, all labels defined in that concept are returned. In this example, a search on *canals* returns documents that contain words such as *canals*, *canal bends*, *canalized streams*, *ditches*, and others. Similarly, a search on *ditches* returns documents containing *ditches*, *canals*, *canal bends*, *canalized streams*, and so on.

xPlore also supports one-way expansion, but only when using an xQuery relationship option. With xQuery, you can specify that relationship “USE” only expands a preferred term with alternative terms, and “UF” only expands alternative terms with a preferred term. For example, the following xQuery would expand *canals* to *canal bends*, *canalized streams*, *ditches*, and so on, but cannot expand *ditches* to *canals* and other terms:

```
.contains text “canals” using thesaurus at “$yourthesaurusuri” relationship “USE”
```

A SKOS thesaurus must use the following RDF namespace declarations:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#">
```

```
<skos:Concept ...>
</skos:Concept>
</rdf:RDF>
```

Terms from multiple languages can be added like the following example:

```
<skos:Concept rdf:about="http://www.fao.org/aos/#canals">
  <skos:prefLabel xml:lang="fr">canal</skos:prefLabel>
  <skos:altLabel xml:lang="fr">coudes du canal</skos:altLabel>
  <skos:altLabel xml:lang="fr">fossés</skos:altLabel>
  ...
  <skos:prefLabel xml:lang="es">canal</skos:prefLabel>
  <skos:altLabel xml:lang="es">curvas del canal</skos:altLabel>
  <skos:altLabel xml:lang="es">zanas</skos:altLabel>
  ...
</skos:Concept>
```

## Importing and viewing a thesaurus

You can view a list of thesauruses for each domain using xPlore Administrator. Click **Diagnostic and Utilities > Thesaurus**. Select the domain (Documentum Server repository) for the thesaurus. The list shows a thesaurus URI, indicates a default thesaurus, and the date it was imported.

To install a thesaurus, select the domain (Documentum Server) for the thesaurus, then browse to a local thesaurus or enter a URI to the thesaurus by providing an XML file in SKOS format. Select *true* for **Default** if this is the default thesaurus for the domain. You can import additional thesauruses for the domain and specify them in DFC or DQL queries.

xPlore registers this library in indexserverconfig.xml as the category *thesaurusdb*. The thesaurus is stored in an xDB library under the domain, for example, root-library/my\_domain/dsearch/SystemInfo/ThesaurusDB. A special path-value index is defined in indexserverconfig.xml for the SKOS format to speed up thesaurus probes. Case-insensitive and space-insensitive lookups in a SKOS dictionary are supported by default.

To modify your thesaurus, update the URI to the thesaurus by providing an XML file in SKOS format. In the XML file, the existing concept will be replaced and a new concept will be added.

## Enabling thesaurus support in a Documentum Server

Enable thesaurus support for Documentum repositories by editing the fulltext engine object configuration for each Documentum Server. Set the value of the **dm\_ftengine\_config** property *thesaurus\_search\_enable*. The following iAPI commands enable thesaurus support. The Documentum Server query plugin and the DFC search service read these values:

```
retrieve,c,dm_ftengine_config
append,c,1,param_name
thesaurus_search_enable
append,c,1,param_value
true
save,c,1
```

## Enabling phrase search in a thesaurus

You can configure the fulltext engine to match entire phrases against the thesaurus. By default, phrase queries are not lemmatized, and the thesaurus is bypassed. The following iAPI command instructs xPlore to match phrases in the thesaurus:

```
retrieve,c,dm_ftengine_config  
append,c,1,param_name  
use_thesaurus_on_phrase  
append,c,1,param_value  
true  
save,c,1
```

## DFC thesaurus APIs

Update DFC in the client application to the latest patch. Use the following APIs to enable thesaurus and to use a custom thesaurus in the DFC search service. These APIs are in IDfSimpleAttrExpression and IDfFulltextExpression. They override the default settings in dm\_ftengine\_config. The *String* argument for setThesaurusLibrary is a URI to the thesaurus that you have imported.

```
void setThesaurusEnabled(Boolean thesaurusSearchEnabled);  
void setThesaurusLibrary(String thesaurusLibrary)
```

The following example enables thesaurus expansion for the metadata *object\_name*.

```
rootExpressionSet.addSimpleAttrExpression(  
    "object_name", IDfValue.DF_STRING, IDfSearchOperation.SEARCH_OP_CONTAINS,  
    false, false, "IIG");  
aSimpleAttrExpr.setThesaurusEnabled(true);
```

## Setting thesaurus support in DQL queries

For all DQL queries, the thesaurus is used only for full-text (SDC) queries and string metadata queries using the *like* or *equals* operator.

If *thesaurus\_search\_enable* is set to true in dm\_ftengine\_config, then the thesaurus is used for DQL queries. To override *thesaurus\_search\_enable* when it is set to false, use the DQL hint *ft\_thesaurus\_search*. For example:

```
select object_name from dm_document search document contains 'test'  
enable(ft_thesaurus_search)
```

To specify a specific thesaurus in a DQL query, use the hint *ft\_use\_thesaurus\_library*, which takes a string URI for the thesaurus. The following example overrides the thesaurus setting in dm\_ftengine\_config because it adds the *ft\_thesaurus\_search* hint. If thesaurus search is enabled, use only the *ft\_use\_thesaurus\_library* hint.

```
select object_name from dm_document search document contains 'test'  
enable(ft_thesaurus_search, ft_use_thesaurus_library ('  
http://search.opentext.com/myDomain/myThesaurus.rdf'))
```

## Setting thesaurus properties in xQuery expressions

xPlore supports the XQuery thesaurus option as specified in the XQuery and XPath Full Text 1.0 specification.

To enable a thesaurus, add *using thesaurus default* to the xQuery expression. For example:

```
object_name ftcontains "IIG" with stemming using stop words default
using thesaurus default entire content
```

The following query enables thesaurus search with a phrase:

```
. ftcontains "programming language" with stemming
using stop words default using thesaurus default
```

## Logging thesaurus actions

Set the log level to DEBUG in xPlore administrator: **Services > Logging > dsearchsearch**. The following information is logged:

- Thesaurus is found from the specified URI. For example:

```
attempt to verify existence of thesaurus URI:
http://search.opentext.com/testenv/skos.rdf
...
successfully verified existence of thesaurus URI:
http://search.opentext.com/testenv/skos.rdf
```

- Argument values that are passed into getTermsFromThesaurus: input terms, relationship, minValueLevel, maxValueLevel. For example:

```
calling getTermsFromThesaurus with terms [leaVe],
relationship null, minValueLevel -2147483648, maxValueLevel
2147483647
```

- Thesaurus XQuery statement. For example:

```
thesaurus lookup xquery: declare option xhive:fts-analyzer-class
'com.emc.documentum.core.fulltext.indexserver.core.index.xhive.
IndexServerAnalyzer';
declare namespace rdf = '
http://www.w3.org/1999/02/22-rdf-syntax-ns#';
declare namespace skos = 'http://www.w3.org/2004/02/skos/core#';
declare variable $terms external;
declare variable $relation external;
doc('/testenv/dsearch/SystemInfo/ThesaurusDB')[xhive:metadata(
.,'uri')='http://search.opentext.com/testenv/skos.rdf']/rdf:RDF/skos:
Concept[skos:prefLabel contains text {$terms} entire content]/
skos:altLabel/text()
```

- Tokens that are looked up in the thesaurus. The query term *leaVe* is rendered case-insensitive:

```
executing the thesaurus lookup query to get related terms for
[leaVe]
...
[Returned token: leave]
...
[Total tokens count for reader: 1]
```

- Query plan for thesaurus XQuery execution. Provide the query plan to OpenText Global Technical Services if you are not able to resolve an issue. For example:

```
thesaurus lookup execution plan: query:6:1:Creating query
plan on node /testenv/dsearch/SystemInfo/ThesaurusDB
query:6:1:for expression ...[xhive:metadata(., "uri") =
http://search.opentext.com/testenv/skos.rdf"]/child:::
{http://www.w3.org/1999/02/22-rdf-syntax-ns#}RDF/child:::
{http://www.w3.org/2004/02/skos/core#}
```

```
Concept[child:::{http://www.w3.org/2004/02/skos/core#}prefLabel
[. contains text terms@0]]/child::
{http://www.w3.org/2004/02/skos/core#}altLabel/child::text()
query:6:1:Using query plan:
query:6:1:index(Concept)
[parent:::{http://www.w3.org/1999/02/22-rdf-syntax-ns#}
RDF[parent::document()][xhive:metadata(., "uri") =
http://search.opentext.com/testenv/skos.rdf"]]/child::
{http://www.w3.org/2004/02/skos/core#}altLabel/child::text()
]
```

- Related terms that are returned from the thesaurus. For example:

```
[related terms from thesaurus lookup query
[Absence from work, Absenteeism, Annual leave, Employee vacations,
Holidays from work, Leave from work, Leave of absence, Maternity
leave, Sick leave]]
```

You can also inspect the final Lucene query. This query is different from the original query because it contains the expanded terms (alternate labels) from the thesaurus. In xPlore Administrator, open **Services > Logging** and expand *xhive*. Change the log level of com.xhive.index.multipath.query to DEBUG. The query is in the xDB log as “generated Lucene query clauses”. xdb.log is in *xplore\_home/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/logs*. The tokens are noted as *tkn*. For example:

```
generated Lucene query clauses(before optimization):
+(((>/dmftmetadata<0>/dm_sysobject<0>/a_is_hidden<0>/ txt:false)^0.0)) +
(((>/dmftversions<0>/iscurrent<0>/ txt:true)^0.0))
+(>/ tkn:shme >/dmftcontents<0>/ tkn:shme >/dmftcontents<0>/dmftcontent<0>/
tkn:shme >/dmftfolders<0>/ tkn:shme >/dmftinternal<0>/
tkn:shme >/dmftinternal<0>/r_object_id<0>/
tkn:shme >/dmftinternal<0>/r_object_type<0>/
tkn:shme >/dmftkey<0>/ tkn:shme >/dmftmetadata<0>/
tkn:shme >/dmftsecurity<0>/
tkn:shme >/dmftsecurity<0>/ispublic<0>/ tkn:shme >/dmftversions<0>/
tkn:shme >/dmftstamp<0>/
tkn:shme) _xhive_stored_payload_:_xhive_stored_payload_
```

## Troubleshooting a thesaurus

Ensure that your thesaurus is in xPlore. You can view a thesaurus and its properties in the xDB admin tool. Navigate to the /xhivedb/root-library/<domain>/dsearch/SystemInfo/ThesaurusDB library. To view the default and URI settings, click the **Metadata** tab.

Ensure that your thesaurus is used. Compare the specified thesaurus URI in the XQuery to the URI associated with the dictionary. View the URI in the xDB admin tool or the thesaurus list in xPlore administrator. Compare this URI to the thesaurus URI used by the XQuery, in dsearch.log. For example:

```
for $i score $s in collection('/testenv/dsearch/Data') /dmftdoc[. ftcontains 'food
products' using thesaurus at 'http://www.opentext.com/skos'] order by $s descending
return $i/dmftinternal/r_object_id
```

If the default thesaurus on the file system is used, the log records a query like the following:

```
for $i score $s in collection('/testenv/dsearch/Data') /dmftdoc[. ftcontains 'food
products' using thesaurus default] order by $s descending return $i/dmftinternal/
r_object_id
```

You can view thesaurus terms that were added to a query by inspecting the final query. Set `xhive.index.multipath.query = DEBUG` in xPlore administrator. Search for `generated Lucene query clauses`.

## 10.2.5 Configuring query lemmatization

xPlore supports search for similar or like terms, also known as lemmatization, by default. To speed indexing and search performance, you can turn off lemmatization for indexing. See “[Configuring indexing lemmatization](#)” on page 133.



**Note:** If you have already indexed documents, the lemmas cannot be removed from the index. They will be found even if you use a phrase search. You must reindex all documents to remove lemmas from the index.

*Lemmatization in XQueries (including DFC 6.7 or higher).*

You can turn on or off lemmatization of individual queries by using the XQFT modifiers *with stemming* or *without stemming*. The XQFT default is *without stemming*. DFC produces XQuery syntax. The DFC default is *with stemming* except for a phrase search.

*Lemmatization in DQL and DFC queries*

The default is *with stemming*. To turn off stemming in DQL or DFC queries, use a phrase search.

## 10.2.6 Configuring search on compound terms

A compound term is composed of two or more simple words as its components. Compound terms are frequent in some languages such as German or Chinese.

By default, searching on a compound term returns results that contain the compound term, all of the components in the same offset, or their alternate lemmas. For example, searching on Heilmittel returns results with heilmittel, heil or heilen, and mittel when they are adjacent.

xPlore has a set of built-in compound terms defined for all supported languages. You can also define your own list of compound terms by creating custom dictionaries for use by CPS. See “[Adding dictionaries to CPS](#)” on page 152. When a custom compound term conflicts with the built-in compound term in terms of its composition, the build-in term overrides the custom one.

You can configure search to query on compound terms in its entirety but not its components for all languages or a specified set of languages.

Configure the search for compound terms by editing `<xplore_home>/config/indexserverconfig.xml` and setting the property named `query-components-from-compound-words-lang-list` under the `search-config` element to one of the following values:

- `"*"`: Searching on a compound term returns any results that contain the compound term, any of its components, or their alternate lemmas for all languages. The search produces a higher number of, but more ambiguous, search results with some of the results being irrelevant. This is also the default behavior when the property is not configured.
- Supported language code such as “de” (German), “zh” (Chinese), or “kr” (Korean): Searching on a compound term returns any results that contain the compound term, any of its components, or their alternate lemmas only for the specified language(s); for other languages, only results containing the compound term proper are returned. Delimit multiple language codes with commas.
- `""(blank)`: Searching on a compound term only returns results that contain the compound term proper for all languages. This yields a higher level of search accuracy at the cost of fewer returned results.

For example, the following setting specifies that only for the Chinese and German languages, searching on a compound term returns results that contain the compound term, any of its components, or their alternate lemmas; for other languages, only results containing the compound term proper are returned:

```
<search-config>
  <properties>
    ...
    <property value="zh,de" name="query-components-from-compound-word-lang-list"/>
    ...
  </properties>
</search-config>
```

xPlore generates alternative forms for components in the root form list. This behavior may cause some compound words hit more documents than expected. To disable this behavior change, set `com.basistech.elab.compounds_in_alt_lemmas` to `false` in the file `cps_context.xml` located in `<xplore_home>/dsearch/cps/cps_daemon`.

```
<property name="com.basistech.elab.compounds_in_alt_lemmas" value="false"/>
```

## 10.2.7 Using query summaries

The summary is a chunk of text from the original indexed document that contains the searched word. Search terms are highlighted in the summary.

The indexing service stores the content of each object as an XML node in `dftxml` called `dmftcontentref`. For all documents in which an indexed term has been found, xPlore retrieves the content node and computes a summary. Based on how they are produced, there are three types of summaries: dynamic summary, static summary and metadata summary.

- Dynamic summary

All of the following must be true in order to have dynamic summaries. If any condition is false, a static summary is generated.

- `query-enable-dynamic-summary` must be set to true
- The result must be within the first X rows defined by the `max-dynamic-summary-threshold` parameter.
- The size of the extracted text must be less than the `extract-text-size-less-than` attribute.
- The query term must appear within the first X characters defined by `token-size` attribute.
- With native xPlore security and the `security_mode` property of the `dm_ftengine_config` object set to BROWSE, you must have at least READ permission to see dynamic summaries.

Dynamic summaries have a performance impact. After the summary is computed, the summary is reprocessed for highlighting, causing a second performance impact.

You can configure dynamic summary processing on the indexing side as well as on the search side. You can also disable dynamic summaries for better performance.

- Static summaryStatic summaries are computed when the summary conditions do not match the conditions configured for dynamic summaries. Static summaries are much faster to compute but less specific than dynamic summaries.
- Metadata summary

To obtain summaries with highlighted metadata, the metadata attributes must be in both the query constraints and the metadata `highlight` attribute list.

### 10.2.8 Configuring query summaries

To configure query summaries:

1. Configure general summary characteristics.
  - a. To turn off dynamic summaries in xPlore Administrator, select **Services > Search Service** and click **Configuration**. Set `query-enable-dynamic-summary` to `false`.

The first  $n$  characters of the document are displayed, where  $n$  is the value of the parameter `query-summary-display-length`.
  - b. To configure the number of characters displayed in the summary, select **Services > Search Service** in xPlore Administrator. Set `query-summary-display-length` (default: 256 characters around the search terms). If no search term is found, a static summary of the specified length from the beginning of the text is displayed, and no terms are highlighted.

- c. To configure the size of a summary fragment, edit `indexserverconfig.xml`. Search terms can be found in many places in a document. Add a property for fragment size to `search-config/properties` (default 64). The following example changes the default value to 32, allowing up to 8 fragments for a display length of 256:

```
<property value="32" name="query-summary-fragment-size"/>
```

2. Configure static summaries in `indexserverconfig.xml`. Specify the elements (Documentum attributes) that are used for the static summary. Set `category-definitions/category/elements-for-static-summary`. The `max-size` attribute sets the maximum size of the static summary. Default: 65536 (bytes).
3. Configure dynamic summary indexing processing
  - a. Configure the maximum size of content that is evaluated for a dynamic summary. Set the maximum as the value of the `extract-text-size-less-than` attribute on the `category-definitions/category/do-text-extraction/save-tokens-for-summary-processing` element. The default: is -1 (all documents). For faster summary calculation, set this value to a positive value. Larger documents return a static summary.
  - b. Configure the number of characters at the beginning of the document in which the query term must appear. If the query term is not found in this snippet, a static summary is returned and term hits are not highlighted. Set the value of the `token-size` attribute on the `category-definitions/category/do-text-extraction/save-tokens-for-summary-processing` element. The default value is 65536 (64K). A value of -1 indicates no maximum content size, but this value negatively impacts performance. For faster summary calculation, set this value lower.
  - c. Configure the maximum number of results that have a dynamic summary. Dynamic summaries require much more computation time than static summaries. Set the value of `max-dynamic-summary-threshold` (default: 50). Additional results have a static summary. If most users do not go beyond the first page of results, set this value to the page size (for example, 10) for better performance.
4. Configure dynamic summary query processing

In addition to configuring text extraction and lemmatization for dynamic summaries as a part of the indexing process, you can further configure dynamic summary computation on the search side without having to reindex the documents.

Edit `indexserverconfig.xml` and add the `query-dynamic-summary-content-text-size` property under the `search-config` element. The value of this property approximately determines in characters the amount of content text and corresponding tokens to be read from disk and considered for computing the dynamic summary for a document.

For example:

```
<search-config>
...

```

```
<property value="256" name="query-dynamic-summary-content-text-size"/>
...
</search-config>
```

The setting above specifies that approximately 256 characters of the content from a document and corresponding amount of tokens will be read from disk and considered for computing the dynamic summary.

Consider setting the value of *query-dynamic-summary-content-text-size* lower in the following scenarios:

- Your disk suffers from poor performance and limited capacity in a multi-user environment and you want to reduce the disk I/O and improve system performance.
- You do not want to shrink the size of the result display of dynamic summary.



**Note:** While improving system performance, this feature may affect the quality of summaries produced for documents, especially in the case of large documents, where only a small portion of the content text is considered for generating summaries. If a search term occurs at the end of a large document, it may not be highlighted in the summary.

### 10.2.8.1 Highlighting query summaries

The search terms, including lemmatized terms, are highlighted within the summary that is returned to the client search application. Wildcard search terms are also highlighted. For example, if the search term is *ran\**, then the word *rant* in metadata is highlighted.

Highlighting does not preserve query context such as phrase search, AND search, NOT search, fuzzy search, or range search. Each search term in the original query is highlighted separately.

To enable summary highlighting for multiple elements, add the following property under *search-config* in *indexserverconfig.xml*:

```
<property value="true" name="query-summary-process-all-highlight-nodes"/>
```

This is a sample query supported by this feature:

```
for $doc in doc('/defaultDomain/dsearch/Data/default')/dmftdoc[. ftcontains
'relational database' with stemming] return <R> <ID>{string($doc/
dmftmetadata//r_object_id)}</ID> <summary> {xhive:highlight(( $doc/
dmftcontents/*/dmftcontentref,$doc/dmftdoc/dmftcustom))}</summary></R>
```

### 10.2.8.2 Highlighting query metadata summaries

You can obtain a text summary with highlighted metadata only if the metadata attributes are in both the query constraints and the metadata highlight attribute list.

To enable metadata highlighting, add a constraint for a metadata field and call this API:

```
IDfQueryBuilder.addMetadataHighlightAttribute(String attrName)
```

To get highlighted metadata, call this DFC API:

```
IDfResultEntry.getMetadataHighlights()
```

This example shows how to get a summary with highlighted metadata:

```
IDfExpressionSet rootExpressionSet = queryBuilder.getRootExpressionSet();
rootExpressionSet.addFullTextExpression("test");
IDfSimpleAttrExpression simpleAttrExpression =
rootExpressionSet.addSimpleAttrExpression("object_name", IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false, "ftdql");
IDfSimpleAttrExpression simpleAttrExpression2 =
rootExpressionSet.addSimpleAttrExpression("r_object_type",
IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false, "dm_document");
IDfSimpleAttrExpression simpleAttrExpression3 =
rootExpressionSet.addSimpleAttrExpression("r_modifier", IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false, "tuser3");
IDfSimpleAttrExpression simpleAttrExpression4 =
rootExpressionSet.addSimpleAttrExpression("owner_name", IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false, "tuser3");
IDfSimpleAttrExpression simpleAttrExpression5 =
rootExpressionSet.addSimpleAttrExpression("subject", IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false, "test");
queryBuilder.addMetadataHighlightAttribute("object_name");
queryBuilder.addMetadataHighlightAttribute("r_object_type");
queryBuilder.addMetadataHighlightAttribute("r_modifier");
queryBuilder.addMetadataHighlightAttribute("owner_name");
queryBuilder.addMetadataHighlightAttribute("subject");

IDfQueryProcessor processor = searchService.newQueryProcessor(queryBuilder,
false);
IDfResultsSet resultsSet = processor.blockingSearch(0);

int resultSize = processor.getResultsSize();
for (int i = 0; i < resultSize; i++)
{
    IDfResultEntry entry = resultsSet.getResultAt(i);
    IDfEnumeration itMetadata = entry.getMetadataHighlights();
    while (itMetadata.hasMoreElements())
    {
        IDfResultMetadataHighlightEntry metadataHighlightEntry =
(IDfResultMetadataHighlightEntry)itMetadata.nextElement();
        System.out.println("Metadata Highlight AttrName: " +
metadataHighlightEntry.getAttributeName());
        IDfEnumeration itTerm = metadataHighlightEntry.getMatchingTerms();
        while (itTerm.hasMoreElements())
        {
            System.out.println("Metadata Highlight Term: " +
itTerm.nextElement());
        }
        System.out.println("Metadata Highlight Summary: " +
metadataHighlightEntry.getSummary());
    }
}
```

### 10.2.8.3 Configuring summary security

By default, users see search results for which they have BROWSE permission if SUMMARY is not selected. If SUMMARY is in the select list, they see only results for which they have READ permission.

To modify the permissions applied to FTDQL and non-FTDQL search summaries, change the security\_mode property of the *dm\_ftengine\_config* object. Use one of the following values:

- BROWSE: Displays all results for which the user has at least BROWSE permission. If the user has BROWSE permission, the summary is blank.
- READ: Displays all results for which the user has at least READ permission.
- SUMMARY\_BASED (default): If SUMMARY is not in the select list, displays all results for which the user has at least BROWSE permission . If SUMMARY is selected, displays results for which the user has at least READ permission.

The following iAPI example sets the summary mode to READ:

```
retrieve,c,dm_ftengine_config
append,c,1,param_name
security_mode
append,c,1,param_value
READ
save,c,1
```

### 10.2.8.4 Configuring parallel summary calculation

Summary calculation usually requires massive processing and may take a long time to return summary results during queries. To improve query performance, you can enable parallel summary calculations to perform multiple summary calculations simultaneously.

Parallel summary calculation is disabled by default. To enable this feature, set the following property under *search-config* in *indexserverconfig.xml* as follows:

```
<property value="true" name="query-parallel-summary-calculation"/>
```

Additionally, you must set the *parallel\_summary\_computing\_enable* of the *dm\_ftengine\_config* object to true in DFC.

Restart the xPlore instance for the change to take effect.

Once enabled, parallel summary calculation lets you take advantage of the new fn:for-each and fn:parse-xml-fragment features in XQuery 3.0 while keeping compatibility with earlier versions of XQuery.



**Note:** Parallel summary calculation may require more CPU and memory.

The following sample provides the root element for the summary processor to process:declare option xhive:fts-analyzer-class 'com.emc.documentum.core.fulltext.indexserver.core.index.xhive.IndexServerAnalyzer'; declare option

```
xhive:ignore-empty-fulltext-clauses 'true'; declare option xhive:index-
paths-values 'dmftmetadata//owner_name,dmftsecurity/acl_name,
dmftsecurity/acl_domain,/dmftinternal/r_object_id'; let $result := for $dm_
doc in collection("/")/dmftdoc[. contains text 'book' ftor 'China' with
stemming] return $dm_doc let $docs := subsequence($result, 1, 351) let $f :=
function ($x as item()) { xhive:highlight($x)} let $summaries := fn:for-
each($docs,$f) for $doc at $i in $docs return <dmrow><object>{$doc/
dmftkey}</object> {fn:parse-xml-fragment($summaries[$i])} </dmrow>
```

Alternatively, you can provide a list of elements for the summary processor to process as shown below:

```
declare option xhive:fts-analyzer-class 'com.emc.
documentum.core.fulltext.indexserver.core.index.xhive.
IndexServerAnalyzer';declare option xhive:ignore-empty-fulltext-clauses
'true';declare option xhive:index-paths-values 'dmftmetadata//owner_name,
dmftsecurity/acl_name,dmftsecurity/acl_domain,/dmftinternal/r_object_id';
let $result := for $dm_doc in collection("/")/dmftdoc[. contains text
'replication' ftor 'Bertha' with stemming] return $dm_doc let $docs :=
subsequence($result, 1, 351) let $f := function ($x as item()) {
xhive:highlight($x/dmftcontents//dmftcontentref, $x/dmftcustom//dmftcontentref)} let $summaries := fn:for-each($docs,$f) for $doc at $i in
$docs return <dmrow><object>{$doc/dmftkey}</object> {fn:parse-xml-
fragment($summaries[$i])}</dmrow>
```

## 10.2.9 Configuring fuzzy search

Fuzzy search can find misspelled words or letter reversals by calculating the similarity between terms using a Lucene algorithm. When a search is detected to be a fuzzy search, xPlore uses the fuzzy search feature of Lucene. Lucene generates a similar words list from the index. The index of the documents is used as a dictionary in xPlore. (The index can be seen as a dictionary in Lucene, and words whose similarity with the search term is more than the specified similarity are returned.)

By default, fuzzy search is not enabled. Fuzzy search is not applied when wildcards are present. Additionally, fuzzy search is only available in XQueries not in DQL queries. DFC clients must be at least version 6.7 to enable fuzzy search.

Fuzzy search does not work if you enable exact phrase match in queries. For information about configuring exact phrase match in queries, see “[About lemmatization](#)” on page 131.

When enabling fuzzy search, a query runs with the search terms as well as a number of similar terms. To reduce the performance impact that such a query can have, we consider that the first character of the search term is correct and we limit the number of similar terms used in the query. For example, searching on *qadministration* would not return results with the term *administration*. Similarly, searching on *explore* would not return results with the term *xplore*.

You can configure the number of leading characters to ignore, the number of similar terms to use in the query, and the supported distance between terms.

Use the following procedure to enable and configure fuzzy search for all queries except DQL.

1. Check your current `dm_ftengine_config` parameters. Use iAPI, DQL, or DFC to check the `dm_ftengine_config` object. First get the object ID, returned by this API command:

```
retrieve,c,dm_ftengine_config
```

2. Use the object ID to get the parameters:

```
? ,c,select param_name, param_value from dm_ftengine_config where  
r_object_id=dm_ftengine_config_object_id
```

3. If the `fuzzy_search_enable` parameter does not exist, use iAPI, DQL, or DFC to modify the `dm_ftengine_config` object. To add a parameter using iAPI in Documentum Administrator, use `append` like the following:

```
retrieve,c,dm_ftengine_config  
append,c,1,param_name  
fuzzy_search_enable  
append,c,1,param_value  
true  
save,c,1
```

4. Change the allowed similarity between a word and similar words, set the parameter `default_fuzzy_search_similarity` in `dm_ftengine_config`. This default also applies to custom fuzzy queries in DFC and DFS for full-text and properties. Set a value between 0 (terms are different by more than one letter) and 1 (default=0.5).

To verify that your fuzzy search setting has been applied, view the query in `dsearch.log`. You should see the following argument in the query with a similarity value that you have set:

```
using option xhive:fuzzy "similarity=xyz"
```

5. Edit the `xdb.properties` file located in the directory `WEB-INF/classes` of the primary instance.

6. Set the `xdb.lucene.fuzzyQueryPrefixLength` property to the number of leading characters that should be ignored when assessing similar terms. Default: 1.

For example, when setting the prefix value to 0, searching `explore` returns `xplore`, but it has a large impact on the performance. Only set it to 0 if the first character is critical to your business. Setting the prefix to a high value improves the performance but similar terms can be omitted and you lose the benefit of the feature.

7. Set the `xdb.lucene.fuzzyTermsExpandedNumber` property to the maximum number of similar terms used in the query. The most similar terms are used. A smaller value improves query response time. Default: 10.

8. Make the same changes in the `xdb.properties` file for all instances.

#### *Fuzzy search in DFC and DFS*

You can enable fuzzy search on individual queries in DFC or DFS. Set fuzzy search in individual full-text and property queries with APIs on `IDfFulltextExpression` and `IDfSimpleAttrExpression`. Use the operators `CONTAINS`, and `EQUALS` for String object types:

- `setFuzzySearchEnabled(Boolean fuzzySearchEnabled)`
- `setFuzzySearchSimilarity(Float similarity)`: Sets a similarity value between 0 and 1. Overrides the value of the parameter `default_fuzzy_search_similarity` in `dm_ftengine_config`.



**Note:** xPlore does not support negative operators, such as `DOES_NOT_CONTAIN`, in fuzzy search.

To disable fuzzy search, set the property `fuzzy_search_enable` in the `dm_ftengine_config` object to `false`.

### 10.2.10 Configuring index type checking

When a query is performed against indexed documents, the query data type (element data type in the XQuery expression) is checked against the index data type (specified in the subpath definition in `indexserverconfig.xml`). The query data type must be compatible with the index data type for the query to be accepted and matched results returned.

You can adjust the level of compatibility between the two sets of data types to fine-tune query performance by setting the `xdb.lucene.strictIndexTypeCheck` value in `xdb.properties`.

Set `xdb.lucene.strictIndexTypeCheck` to `False` to apply less strict data type checking to queries to improve query performance. This is the default value. When `xdb.lucene.strictIndexTypeCheck` is `False`, xPlore considers index data type string compatible with query data types `dateTime` and `integer` (only in exact match) when handling queries.

The following table shows compatible query and index data type pairs (indicated as Yes) when `xdb.lucene.strictIndexTypeCheck` is set to `False`.

Query Type	Index Type							
	string	integer	double	date	dateTime	time	float	long
string	Yes							
integer	Yes	Yes	Yes				Yes	Yes
double			Yes					
date				Yes	Yes			
dateTime	Yes				Yes			

Query Type	Index Type							
	string	integer	double	date	dateTime	time	float	long
time					Yes	Yes		
float			Yes				Yes	

When `xdb.lucene.strictIndexTypeCheck` is True, a stricter index typing checking rule is enforced. This may cause lower query performance if you did not specify index data types when creating subpath definitions.

The following table shows compatible query and index data type pairs (indicated as Yes) when `xdb.lucene.strictIndexTypeCheck` is set to True.

Query Type	Index Type							
	string	integer	double	date	dateTime	time	float	long
string	Yes							
integer		Yes	Yes				Yes	Yes
double			Yes					
date				Yes	Yes			
dateTime					Yes			
time					Yes	Yes		
float			Yes				Yes	

### Example 10-1:

For example, the following elements in a Documentum Server document are declared as type integer and dateTime respectively:

```
<owner_permit dmfttype="dmint">7</owner_permit>
<r_creation_date dmfttype="dmdate">2010-09-27T22:54:48</r_creation_date>
```

However, without an explicit subpath definition in `Indexserverconfig.xml`, both element values are indexed as string values in multi-path indexes.

Performing the following queries will return different results depending how you set the `xdb.lucene.strictIndexTypeCheck` value:

```
/dmftdoc[dmftmetadata//owner_permit = xs:Integer("7")]
/dmftdoc[dmftmetadata//r_creation_date = xs:dateTime("2010-09-27T22:54:48")]
```

- If `xdb.lucene.strictIndexTypeCheck` = True, the elements will not be returned since the query data types integer and dateTime do not match the index data type string.

- If `xdb.lucene.strictIndexTypeCheck = False`, the element will be returned since the query data types `integer` and `dateTime` are considered compatible with the index data type `string`.



## 10.3 Configuring a wildcard search

Wildcards are used to match candidate terms for further index querying. You can configure wildcard searches to find matching terms in a single term or multi-term search. xPlore supports XQuery specifications for wildcard searches, with the following query patterns:

- “.” matches a single arbitrary character.
- “.?” matches either no characters or one character.
- “.\*” matches zero or more characters.
- “.+” matches one or more characters.
- A period followed by a sequence of characters that match the regular expression `{[0-9]+,[0-9]+}` matches the following: a number of characters, where the number is no less than the number represented by the series of digits before the comma, and no greater than the number represented by the series of digits after the comma.



**Note:** UI, DQL, or DFC may use different characters for wildcards, but these are translated into the supported XQuery patterns.

Wild card search configuration can affect query performance, with complex searches and sub-searches resulting in slower queries. There are two types of wildcard queries. Prefix wildcard queries can utilize a dictionary to easily match terms. Regex wildcard queries apply regex to evaluate and match terms one by one, resulting in slower performance than prefix queries. See [“Configuring a leading-wildcard subpath” on page 269](#).

Slow wildcard searches may result from the following:

- There are many terms to match, especially for regex wildcard queries.
- There are many matched terms, resulting in many sub searches.
- The wildcard search causes an unselective search result. If upper filter logic is required, such as a security filter, then search speed decreases.
- A large number of matched terms requires more memory to calculate relevance. This can result in peaking of the memory footprint, which slows down wildcard search and other activities.

Limitations of wildcard searches:

- Stemming options are not generated by the DFC and DQL plugin. For example, in a wildcard search “drove ca.\*” with wildcards”, ‘drove’ will not be able to

match its lemma ‘drive’. However, xPlore supports the XQuery standard and can support both options at the same time. For example:

```
for $i score $s in /dmftdoc[. ftcontains 'drove ca.*' with wildcards with
stemming] order by $s descending return <d>{$i/dmftmetadata//r_object_id}
{ $i/dmftmetadata//object_name } { $i/dmftmetadata//r_modifier }</d>
```

- Tokenization is not applied for languages that do not depend on whitespace to separate terms. For example, in a wildcard search “□江大.\*” with wildcards”, it is not possible to match ‘□江’ and ‘大.\*’, only ‘□江大□’, if it is indexed as one term.
- A combination of wildcard search and negative operators (does not contain, does not equal, not in) is not supported by the search engine in most cases. Term cutoffs can cause partial results, and can cause incorrect results when a negative operator is applied. Without term cutoff, performance issues increase. The only exception is when the following conditions are met:
  - xdb.lucene.prefixQueryCutoff=false. See “[Limiting wildcards and common terms in search results](#)” on page 270.
  - The query can be converted to a simple prefix query and is not in a phrase query.
- A search with only wildcards will cause an exception, by default. In most cases, such a search is meaningless and causes very poor performance. If you need to run this type of search, make the following changes:
  1. In indexserverconfig.xml, add *search-config* property
 

```
<property value="false"
name="query-discard-unselective-wildcard-query" />
```
  - By default, true.
  2. In xdb.properties, add option *xdb.lucene.noTokenFtcontainsToMatchAll=true*. By default, false.

### 10.3.1 Configuring a leading-wildcard subpath

Since prefix wildcard queries are faster than regex queries, xPlore provides the leading-wildcard subpath option. This subpath optimizes the regex wildcard query by converting it to a prefix query.

For example, a custom attribute *customer* has a variable prefix such as SrcAtlantic or ResAtlantic and the user wants to perform a search specifying \*Atlantic. To support this custom metadata leading wildcard, do the following:

1. Add a subpath entry to indexserverconfig.xml:

 **Example 10-2:**

```
<sub-path leading-wildcard="true" value-comparison="true"
path="dmftmetadata//customer" />
```



2. To apply the changes, rebuild the index.



**Note:** This optimization results in increased disk space usage.

### 10.3.2 Limiting wildcards and common terms in search results

Matched terms of a wildcard search have a cutoff to limit resource usage. In addition to configuring the cutoff, you can configure a frequency limit so that common terms that appear often, such as stop words, are dropped from the search.

Open `xdb.properties` in the directory `WEB-INF/classes` of the specified instance. If the following keys do not exist, you can add them and then restart `xPlore` to implement the changes:

- `xdb.lucene.termsExpandedNumber`: Sets the cutoff. Default: 65536. The value must be less than `xdb.lucene.maxBooleanClause`.
- `xdb.lucene.ratioOfMaxDoc`: Sets the maximum term frequency in the index. Default: 1. Term frequency is recorded during indexing. For example, if the ratio is 0.5 or higher, the query does not search for terms that occur in more than half of the documents in the index. A search for `*nd` would reject hits for `and` because it occurs in more than half of the documents.
- `xdb.lucene.prefixQueryCutoff`: Specifies whether or not to enable cutoff for prefix queries (e.g. `'abc.*'`). Disabling cutoff may result in slower performance if there is a very large number of matching terms. Default: true.

To get cutoff messages through DFC, enable them with the following API:

```
IDfQueryBuilder.setCutoffMessageRetrieved(true)
```

Retrieve cutoff messages by calling the following DFC API. Each docbase source may return a cutoff message. The return result is a map from docbase source names to cutoff messages.

```
IDfQueryProcessor.getCutoffMessages()
```

This example shows how to get cutoff messages from DFC:

```
IDfQueryBuilder queryBuilder = queryMgr.newQueryBuilder("dm_document");
queryBuilder.addSelectedSource(docbase);
queryBuilder.addResultAttribute("r_object_id");
queryBuilder.setCutoffMessageRetrieved(true);

IDfExpressionSet rootExpressionSet = queryBuilder.getRootExpressionSet();
rootExpressionSet.addFullTextExpression("a*");
IDfQueryProcessor processor = searchService.newQueryProcessor(queryBuilder,
false);
IDfResultSet resultsSet = processor.blockingSearch(0);

System.out.println("resultsSet.size(): " + resultsSet.size());
Map<String, String> cutoffMessages = processor.getCutoffMessages();
for (Map.Entry<String, String> entry : cutoffMessages.entrySet())
{
    System.out.println(entry.getKey() + "-->" + entry.getValue());
```

### 10.3.3 Supporting wildcard searches in DFC

To let DFC generate wildcard queries, you can either use the default mode and enter explicit wildcard characters, or you can customize the wildcard mode.

For simple full-text searches, including Webtop simple search, Webtop Advanced search Contains field, and global search in xCP 2.0, you can modify the ft\_wildcards\_mode in the dm\_ftengine\_config object. Valid values are:

- none: wildcard is treated as a literal character.
- explicit: wildcard character must be entered (default)
- implicit: \* is added to the beginning and end of every search term.
- trailing\_implicit: \* is added to the end of every search term.

To modify the *dm\_ftengine\_config* object in the Documentum Server, do the following:

1. Using iAPI in Documentum Administrator or Server Manager, get the object ID:

```
retrieve,c,dm_ftengine_config where
object_name like 'DSearch%'
```

2. Use the object ID to get the dm\_ftengine\_config parameters and values. In the following example, the value of r\_object\_id returned in step 1 is used to get the parameters:

```
? ,c,select param_name, param_value from dm_ftengine_config
where r_object_id='080a0d6880000d0d'
```

3. If the wildcards configuration parameters are not returned, configure them. Append *aparam\_name* and *param\_value* element and set its value. For example:

```
retrieve,c,dm_ftengine_config
append,c,1,param_name
ft_wildcards_mode
append,c,1,param_value
explicit
save,c,1
```

4. To change an existing parameter, locate the position of the *param\_name* attribute value of the parameter. Use *set* as follows:

```
retrieve,c,dm_ftengine_config
dump,c,1 //locates the position
set,c,1,param_value[i] //position of ft_wildcards_mode
implicit
save,c,1
```

For metadata searches, modify contains,starts with/ends with>equals in the following parameters:

- *metadata\_contains\_wildcards\_mode*: separately controls metadata search for *contains* operator. Valid values: none | explicit(default) | implicit | trailing\_implicit.

- *metadata\_startswith\_wildcards\_mode*: separately controls metadata search for *startswith* operator. Valid values: none | explicit(default) | implicit. With implicit, the wildcard is added at the end of the search term.
- *metadata\_endswith\_wildcards\_mode*: separately controls metadata search for *endswith* operator. Valid values: none | explicit(default) | implicit. With implicit, the wildcard is added at the beginning of the search term.
- *metadata\_equals\_wildcards\_mode*: separately controls metadata search for *equals* operator. Valid values: none | explicit(default).

### 10.3.4 Supporting wildcard searches in DQL

The *fastWildcardCompatible* property is available in `dm_ftengine_config` to support searches for word fragments (FAST wildcard behavior). By default false, wildcards are treated as whitespace or word separator. To support wildcards in all DQL searches, change *fastWildcardCompatible* to true. To support wildcards in a single term search, use the DQL hint `FT_CONTAIN_FRAGMENT`.



**Note:** Documentum Server users may be more familiar with the older term fragment search, which is the same concept as wildcard search.

## 10.4 Configuring Documentum search

- “Limiting query fetch time” on page 272
- “Query plugin configuration (`dm_ftengine_config`)” on page 273
- “Making types and attributes searchable” on page 274
- “Running folder descend queries” on page 275
- “DQL, DFC, and DFS queries” on page 276
- “Routing a query to a specific collection” on page 317
- “Tracing Documentum queries” on page 280

### 10.4.1 Limiting query fetch time

Sometimes, a query may take a long time – maybe hours – to return any results, which unnecessarily occupies system resources during the process. You can limit the query fetch time to avoid this.

To limit the query fetch time, add the property `query-max-fetch-time` in `indexserverconfig.xml` and specify in milliseconds the maximum amount of time allowed to pass before the query times out. Set an integer value greater than 0; for example:

```
<search-config>
  <properties>
    ...
    <property name="query-max-fetch-time" value="10000" />
    ...
  </properties>
</search-config>
```

```
</properties>
</search-config>
```

The default, this feature is not enabled.

To troubleshoot slow queries, see “[Troubleshooting slow queries](#)” on page 305.

## 10.4.2 Query plugin configuration (dm\_ftengine\_config)

The Documentum Server query plugin settings are set during index agent installation. Change them only if the Documentum Server or xPlore environment changes. The following settings affect query processing. If you change them, you do not need to restart the Documentum Server. For a full description of dm\_ftengine\_config attributes, see “[dm\\_ftengine\\_config](#)” on page 397.

Queries fail if the wrong (FAST) query plugin is loaded in the Documentum Server. Check the Documentum Server log after you start the Documentum Server. The file *repository\_name.log* is located in <DOCUMENTUM\_HOME>/dba/log. Look for the line like the following. It references a plugin with DSEARCH in the name.

```
[DM_FULLTEXT_T_QUERY_PLUGIN_VERSION]info: "Loaded FT Query Plugin:
.../DSEARCHQueryPlugin.dll...FT Engine version: X-Hive/DB 10"
```

To check your current dm\_ftengine\_config settings, use iAPI, DQL, or DFC. To view existing parameters using iAPI in Documentum Administrator:

- First get the object ID:

```
retrieve,c,dm_ftengine_config
... <dm_ftengine_config_object_id>
```

- Use the object ID to get the parameters:

```
? ,c,select param_name, param_value from dm_ftengine_config
where r_object_id=dm_ftengine_config_object_id
```

- Add a missing parameter using iAPI *append* like the following:

 **Example 10-3:**

```
retrieve,c,dm_ftengine_config
append,c,1,param_name
acl_check_db
append,c,1,param_value
T
save,c,1
```



- To change an existing parameter, use *set* like the following:

 **Example 10-4:**

```
retrieve,c,dm_ftengine_config
set,c,1,param_name
acl_check_db
set,c,1,param_value
```

```
false  
save,c,1
```



3. To remove an existing parameter, use *remove* instead of *set*.

When setting the values of the *param\_name* and *param\_value* elements using iAPI, you must remove any redundant space at the beginning or end of the value. Otherwise, the setting fails. For example, the following wildcard setting fails because of the space before *ft\_wildcards\_mode*.

```
retrieve,c,dm_ftengine_config  
append,c,1,param_name  
  ft_wildcards_mode  
append,c,1,param_value  
explicit  
save,c,1
```

Redundant spaces before or after parameter names or values can cause unexpected errors. The following DQL statement helps you locate redundant spaces in the *dm\_ftengine\_config* object:

```
select param_name, param_value from dm_ftengine_config  
where any param_name like '% ' or any param_name like ' %'  
or any param_value like '% ' or any param_value like ' %'enable(row_based);
```

### 10.4.3 Making types and attributes searchable

You can create or alter types to make them searchable and configure them for full-text support. For information on making attributes non-searchable, see “[Making metadata non-searchable](#)” on page 107.

#### *Allowing indexing*

The *a\_full\_text* attribute is defined for the *dm\_sysobject* type and subtypes (default: true). Content and properties are indexed when a Save, Saveasnew, Checkin, Destroy, Branch, or Prune operation is performed on the object. When *a\_full\_text* is false, only the properties are indexed. Users with Sysadmin or Superuser privileges can change the *a\_full\_text* setting using Documentum Administrator.

#### *Setting indexable formats*

Properties of the format object (type *dm\_format*) 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 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 indexing by creating a rendition in an indexable format. For more details, see *OpenText Documentum Search Development Guide*.

#### *Allowing search*

Set the *is\_searchable* attribute on an object type to allow or prevent searches for objects of that type and its subtypes (default: true). Valid values: 0 (false) and 1 (true). The client application must read this attribute. If *is\_searchable* is false for a type or attributes, Webtop does not display them in the search UI.

### *Lightweight sysobjects (LWSOs)*

Lightweight sysobjects group the attribute values that are identical for a large set of objects. This redundant information is shared among the LWSOs from the shared parent object. For LWSOs like *dm\_message\_archive*, the client application must configure searchable attributes. Use CREATE TYPE and ALTER TYPE FULLTEXT SUPPORT switches to specify searchable attributes. For more information on this configuration, see *OpenText Documentum Server DQL Reference Guide*. For information on supporting extended search with LWSOs, see *OpenText Documentum Search Development Guide*.

### *Aspects*

Properties associated with aspects are not indexed by default. If you wish to index them, use an ALTER ASPECT statement to identify the aspects you want indexed. For more information on this statement, see *OpenText Documentum Server DQL Reference Guide*.

## 10.4.4 Running folder descend queries

Folder descend query performance depends on the folder hierarchy (number of folders) and data distribution across folders. The following conditions can degrade query performance:

- Many descending folders in the target folder, but small *folder\_cache\_limit*.  
If the number of descending folders is greater than the *folder\_cache\_limit* configured in the *dm\_ftengine\_config* object, folder condition is not evaluated with other conditions by the Lucene index. However, folder condition is applied to the Lucene index result set based on other conditions, so documents are filtered sequentially. To improve query performance, increase the *folder\_cache\_limit*.
- Many descending folders and large *folder\_cache\_limit*, but low memory capacity.  
With a large *folder\_cache\_limit*, xPlore requires more memory, or a larger heap, for folder conditions to be evaluated along with other conditions by the Lucene index. A very large *folder\_cache\_limit* is not recommended, since this may cause out-of-memory errors or memory vibration when many queries are running. To stabilize the system, decrease the *folder\_cache\_limit* or limit concurrent queries, or add more memory to xPlore.



**Note:** You may need to adjust volume and query loads to find an appropriate *folder\_cache\_limit*.

You can detect the folder descend problem in the timeout stack trace. The timeout occurs for the FilterFoldersFunction:

```
2015-01-18 16:48:38,451 WARN [pool-12-thread-9]
com.emc.documentum.core.fulltext.search - Xhive exception message:
INTERRUPTED com.xhive.error.XhiveInterruptedException: INTERRUPTED
... at com.emc.documentum.core.fulltext.indexserver.services.folders.
FilterFoldersFunction.executeXQuery(FilterFoldersFunction.java:86)
```

Set the *folder\_cache\_limit* in the *dm\_ftengine\_config* object to the expected maximum number of folders in the query (default = 2000). If the folder descend condition

evaluates to less than the `folder_cache_limit` value, then folder IDs are pushed into the index probe, making the query much faster. If the condition exceeds the `folder_cache_limit` value, the folder constraint is evaluated separately for each result.

You can apply FilterFoldersFunction performance improvements included in the new setup by upgrading to the latest xPlore version and performing the following steps:

1. In `Indexserverconfig.xml`, create the following subpath definition:

```
<sub-path path="dmftfolders/i_folder_id" full-text-search="false"  
value-comparison="true" returning-contents="true"/>
```

2. Rebuild the index.

#### 10.4.5 DQL, DFC, and DFS queries

DFC-based client applications use the DFC query builder package to translate a query into an XQuery statement. DFS similarly generates XQuery statements. If your application issues DQL queries, the Documentum Server query plugin for xPlore translates the DQL into an XQuery expression. Any part of the query that is not full-text compliant (NOFTDQL) is evaluated in the Documentum Server database, and the results are combined with results from the XQuery.



**Note:** XQueries generated by DFC and the Documentum Server query plugin may return different ranking scores.

You can also turn off XQuery generation to force DQL for one of the following use cases:

- To evaluate security on the Documentum Server security. By default, search results are filtered for security in xPlore before they are returned to the Documentum Server, which has much better performance.
- To get the most recent metadata updates before they are indexed.



**Note:** When XQuery generation is turned off, search performance is worse. Many search features are not supported in DQL as described in the following table.

**Table 10-2: Differences between DQL and DFC/DFS queries**

DQL	DFC and DFS
Returned attribute values are read from the database. No latency when objects are updated.	Returned attribute values are read from xPlore. Short latency (in the minute range) between Documentum Server and xPlore.
No latency for security evaluation	Short latency (in the minute range) for security updates but faster search results
No VQL equivalent	Extended object search (VQL-type support). See “ <a href="#">Rewriting VQL queries</a> ” on page 251.

DQL	DFC and DFS
No facets	Out-of-the-box and custom facets
No hit count	Hit count
Hints file supported	Hints file not supported unless XQuery generation is turned off.
Fragment search	Fragment search supported (by default, wildcard must be explicit) and configurable
No fuzzy search	Fuzzy search supported and configurable
By default, sorting on attribute is performed by the database on results returned from xPlore. You can alter the behavior by adding dm_ft_order_by_enabled to dm_docbase_config as shown below:  <pre> 1 retrieve,c,dm_docbase_config 2 append,c,l,r_module_name 3 dm_ft_order_by_enabled 4 append,c,l,r_module_mode 5 save,c,l 6 reinit,c </pre> <p>Ensure that the subpath is configured correctly in indexserverconfig.xml. For example:</p> <pre> set "sortable= true" for subpath "dmftmetadata//object_name" </pre> <p>Then, using DQL, you can sort by attribute object_name. For example:</p> <pre> select r_object_id from dm_sysobject search document contains 'john' order by object_name </pre> <p>This is applicable for sorting results supported in xPlore with the appropriate index configuration.</p>	Sorting results supported in xPlore with the appropriate index configuration.
No paging	Paging of results
Stemming	Stemming can be disabled at the DFC level for EQUALS or NOT_EQUALS constraints, to return only exact matches
Thesaurus support only on the full-text constraint	Thesaurus support on both full-text and metadata constraints

### 10.4.6 Documentum Server and DFC client search differences

Support for new search features has been added progressively to versions of Documentum Server. The following table illustrates the supported features by Documentum Server and DFC version. On all but the most recent version of Documentum Server or DFC, these features require latest hotfix.

DFC 6.6 search service and later generates XQuery unless you set one of the following:

- `dfc.search.xquery.generation.enable` is *false*
- `ftsearch_security_mode` in `dm_ftengine_config` is *0*

**Table 10-3: Supported search features**

Feature	Documentum Server	DFC on client
DFC search service generates DQL query.		6.5 SP2, SP3. (6.6 and higher if XQuery generation is turned off.)
DFC search service generates XQuery.		6.6, 6.7.x unless XQuery generation is turned off.
Facets	No dependency	6.6 or higher
Thesaurus	6.5 SP2, SP3, 6.6, 6.7.x	6.5 SP2, SP3, 6.6, 6.7.x (DQL), 6.7.x (DQL and XQuery)
Fuzzy search	6.7.x	6.7.x XQuery and DQL
Query subscription	6.7 SP1	6.7 SP1
Extended object search		6.6 or higher
Wildcards in metadata		6.6 or higher
API for exact match		6.7.x XQuery only
Debugging enhancement		6.7.x

### 10.4.7 DQL Processing

The DFC and DFS search services by default generate XQuery expressions, not DQL, for xPlore. DQL hints in a hints file are not applied. You can turn off XQuery generation in `dfc.properties` so that DQL is generated and hints are applied. Do not turn off XQuery generation if you want xPlore capabilities like facets.

If query constraints conform to FTDQL, the query is evaluated in the full-text index. If all or part of the query does not conform to FTDQL, only the SDC portion is evaluated in the full-text index. All metadata constraints are evaluated in the Documentum Server database, and the results are combined.

The following configurations turn off XQuery and render a query in DQL:

- `dfc.search.xquery.generation.enable = false` in `dfc.properties`
- `ftsearch_security_mode` is 0. See “[Changing search results security](#)” on page 63.
- `acl_check_db` is true. See “[Changing search results security](#)” on page 63.

## Enabling DQL hints (turn off XQuery)

The Webtop search components use the DFC query builder package to construct a query. The DFC query builder adds the DQL hint TRY\_FTDQL\_FIRST. This hint prevents timeouts and resource exceptions by querying the attributes portion of a query against the repository database. The query builder also bypasses lemmatization by using a DQL hint for wildcard and phrase searches.

For information on using a hints file, see *OpenText Documentum Search Development Guide*.

To use a DQL hints file or hints in a DQL query, disable XQuery generation by DFC or DFS. The hints file allows you to specify certain conditions under which a database or standard query is done in place of a full-text query. Turn off XQuery generation by adding the following setting to `dfc.properties` on the DFC client application:

```
dfc.search.xquery.generation.enable=false
```

## Unsupported DQL

xPlore does not support the DQL SEARCH TOPIC clause or pass-through DQL.

## Comparing DQL and XQuery results

DQL and XQuery results can be compared by testing a query with the DFC search service. First, run an XQuery expression (default). Next, turn off XQuery generation and generate DQL.

Sometimes you see different results if you query for metadata on an object that has not yet been indexed. A query in DQL returns results directly from the repository. A query in XQuery does not return results until xPlore has updated the index.

## DQL hints migration

The following table lists the DQL hint support in xPlore. For a full description of the referenced DQL hints, see *Including DQL hints* in *OpenText Documentum Server DQL Reference Guide*.

**Table 10-4: DQL hint migration for xPlore**

DQL	xPlore
RETURN TOP N	Not needed by xPlore. DFC search service asks for the exact number of results.

DQL	xPlore
FT_CONTAIN_FRAGMENT	Default is FT_CONTAIN_WORD. This setting is deprecated. Modify ft_engine_config to support fragments.
ENABLE(dm_fulltext('qtf_lemmatize=0 1'))	Turn on DQL generation in dfc.properties or turn off lemmatization in xPlore indexserverconfig.xml.
FOR READ/BROWSE/DELETE/MODIFY option	Turn on DQL generation in dfc.properties. (Not a hint, but used to configure queries in DQL hint manager.)
FT_COLLECTION	Use DFC query builder addPartitionScope() or turn on DQL generation
TRY_FTDQL_FIRST, NOFTDQL	Turn on DQL generation in dfc.properties.
FTDQL	Use default XQuery generation unless other hints are added
ROW_BASED or database-specific hints	No equivalent
All other hints	Turn on DQL generation in dfc.properties.

#### 10.4.8 Tracing Documentum queries

You can trace queries using the MODIFY\_TRACE apply method. On Windows, this command controls tracing for all sessions. On Linux, tracing is session-specific. There are four possible levels of tracing queries in Documentum environments. You can trace subsystems with one of the following values:

- all  
Traces everything (sum of cs, ftplugin, and ftengine).
- cs  
Traces Documentum Server search operations such as initializing full-text in-memory objects and the options used in a query.
- ftplugin  
Traces the query plugin front-end operations such as DQL translation to XQuery, calls to the back end, and fetching of each result.
- ftengine  
Traces back-end operations: HTTP transactions between the query plugin and xPlore, the request stream sent to xPlore, the result stream returned from xPlore, and the query execution plan.
- none

##### Turning tracing on or off

Use the iAPI command to turn off tracing:

```
apply,c,NULL,MODIFY_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,none
```

Use the iAPI command to turn on tracing:

```
apply,c,NULL,MODIFY_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,all
```

## Tracing in the log

Trace messages are written to <DOCUMENTUM\_HOME>/dba/log/fulltext/fttrace\_<repository\_name>.log. The log entry contains the following information:

- Request query ID, so that you can find the translated query in the xPlore fulltext log (<DOCUMENTUM\_HOME>/dba/log/fulltext/fttrace\_repository\_name.log).
- The XQuery that was translated from DQL
- The request and response streams, to diagnose communication errors or memory stream corruption
- *dm\_ftengine\_config* options
- The query execution plan is recorded in the xplore log dsearch.log in the logs subdirectory of the WildFly deployment directory. You must set trace to *ftengine* or *all*.

# 10.5 Supporting subscriptions to queries

## 10.5.1 About query subscriptions

- “Overview” on page 281
- “Query subscription creation” on page 282
- “Query subscription execution” on page 283
- “Subscribing Queries Running on Multiple Repositories” on page 284

### Overview

Query subscriptions is a feature in which a user can:

- Specify to automatically run a particular saved search (full-text or metadata-only) at specified intervals (once an hour, day, week, or month) and return any new results.

The results can be discarded or saved. If the results are saved, they can be merged with or replace the previous results.

- Unsubscribe from a query.
- Retrieve a list of their query subscriptions.
- Be notified of the results via a *dmi\_queue\_item* in the subscribed user Inbox and, optionally, an email.

- Execute a workflow, for example, a business process defined in xCP.

Query subscriptions run in Documentum Server 6.7 SP1 or later with DFC 6.7 SP1 or later. Support for query subscriptions is installed with the Documentum Server. A DFC client like Webtop or CenterStage must be customized using DFC 6.7 SP1 or later to present query subscriptions to the user.

Because automatically running queries at specified intervals can negatively affect xPlore performance, tune and monitor query subscription performance.

## Query subscription creation

When a user subscribes to a query, the following objects are created:

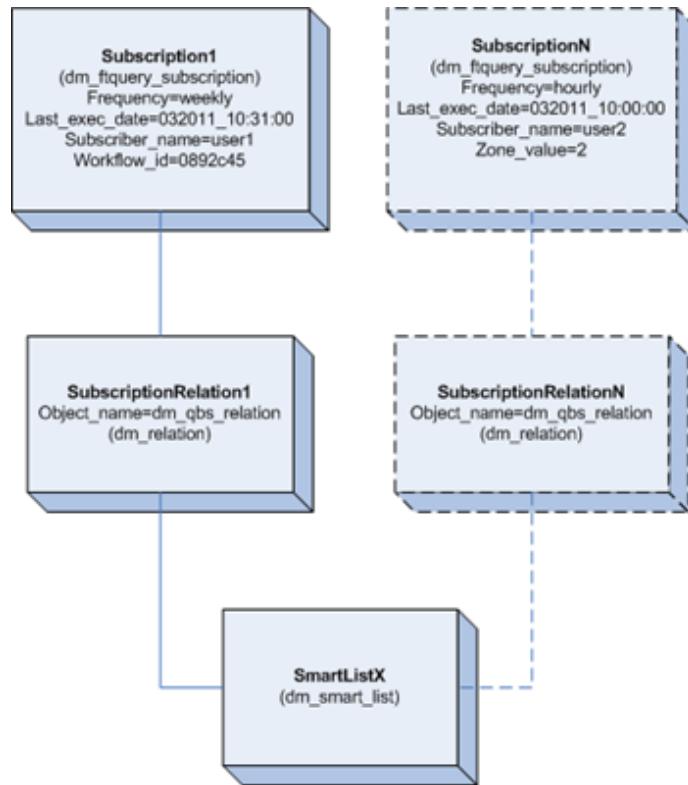
- A dm\_relation object relates the dm\_smart\_list object to a single dm\_ftquery\_subscription object.
- A dm\_ftquery\_subscription object specifies the attributes of the subscription and the most recent query results.  
A user can subscribe to a dm\_smart\_list object only once.



**Note:** Queries are saved as dm\_smart\_list objects through the DFC Search Service API or any user interface that exposes that API like Webtop or Taskspace Advanced Search.

A dm\_smart\_list object contains the query.

**Figure 10-2** illustrates how the different objects that comprise query subscriptions are related. A query can be subscribed to multiple times. A single dm\_smart\_list object can have multiple dm\_relation objects, and each single dm\_relation object, in turn, is related to a single dm\_ftquery\_subscription object. For example, both *Subscription1* and *SubscriptionN* are related to the same dm\_smart\_list, *SmartListX*, but through different dm\_relation objects, *SubscriptionRelation1* and *SubscriptionRelationN*, respectively. Furthermore, *Subscription1* and *SubscriptionN* have different characteristics. For example, *Subscription1* is executed once a week whereas *SubscriptionN* executes once an hour. There is only one subscriber per subscription; that is, the subscriber of *Subscription1* is user1 and the subscriber for *SubscriptionN* is user2.



**Figure 10-2: Query Subscription Object Model**

## Query subscription execution

When one of four pre-installed jobs run, the following sequence of actions occurs:

1. Matching query subscriptions are executed sequentially.

A matching `dm_ftquery_subscription` object is one that has a frequency attribute value that matches the job method's `-frequency` value. For example, a job method frequency value of hourly executes all matching `dm_ftquery_subscription` objects once an hour.



**Note:** All queries run under the subscriber user account.

2. One of the following conditions occurs:

- a. If new results are found, then the new results are returned and one of the following occurs:
  - (Default) A `dmi_queue_item` is created and, optionally, an email is sent to the subscribing user.
  - A custom workflow is executed. If the workflow fails, then a `dmi_queue_item` describing the failure is created.



**Note:** You must create this workflow.

- b. If no new results are found, then the next matching query subscription is executed.
3. Depending on the *result\_strategy* attribute value of the dm\_ftquery\_subscription object, the new results:

- Replace the current results in the dm\_ftquery\_subscription object.
- Merge with the current results in the dm\_ftquery\_subscription object.
- Are discarded.



**Note:** The number of results returned per query as well as the total number of results saved are set in the dm\_ftquery\_subscription object *max\_results* attribute.

4. The next matching query subscription is executed.
5. After all matching query subscriptions have been executed, the job stops and a job report is saved.



**Note:** If the stop\_before\_timeout value (the default is 60 seconds) is reached, then the job is stopped and any remaining query subscriptions are executed when the job runs next time.

## Subscribing Queries Running on Multiple Repositories

Starting from xPlore 1.5, you can perform cross-repository searches in query-based subscriptions. To use this feature, the java method server must be a trusted client. If the Documentum Server and java method server are deployed on the same host, the java method server is trusted client by default. If the java method server is set up separately, you must turn on the trusted client configuration in DA.

Compared with subscriptions to queries running on single repository, subscribing cross-repository queries requires the following additional configurations:

1. In DA, add two arguments *-repo\_names* and *-repo\_owners* to the query-based subscription job to list all target repositories the query runs on and the corresponding repository owner names respectively. For example: *-repo\_names: repoA, repoB, repoC -repo\_owners: dmadmin, dmadmin, dmadmin2*
2. Create a user account for the subscriber in each target repositories. If the subscriber name doesn't exist in one repository, the repository is ignored.

When the query-based subscription job is running, the subscriber can receive the search result in their inbox. The returned result set is composed of the query results from all target repositories.

Note that in cross-repository searches, the argument *-max\_results* stands for the maximum number of query results that can be returned from one repository instead of the maximum number of query results for the whole result set. For example, if

you run a query on three repositories with `-max_results` set to 10, you can receive 30 results at most.

### 10.5.2 Installing the query subscription DAR

If you have installed Documentum Server 6.7 SP1 or higher, you do not need to install the query subscription DAR.

The installation files are located in `xplore_home/setup/qbs`.

The query subscription DAR file contains:

- An SBO that provides functionality to subscribe, unsubscribe, and list the current subscriptions for a user.
  - A TBO that provides functionality to run the saved search query and save results.
  - `dm_ftquery_subscription` object type
  - `dm_qbs_relation` object (`dm_relation_type`)
  - Jobs and an associated Java class that runs the subscribed queries.
  - A query subscription test program with which the administrator can validate that query subscriptions were set up properly.
1. Copy `qbs.dar`, `DarInstall.bat` or `DarInstall.sh`, and `DarInstall.xml` from `xplore_home/setup/qbs` to a temporary install directory.
  2. Edit `DarInstall.xml`:
    - a. Specify the full path to `qbs.dar` including the file name, as the value of the `dar` attribute.
    - b. Specify your repository name as the value of the `docbase` attribute.
    - c. Specify the repository superuser name as the value of the `username` attribute.
    - d. Specify the repository superuser password as the value of the `password` attribute.



#### Example 10-5:

For example:

```
<emc.install dar="C:\Downloads\qbs.dar" docbase="xplore1"
username="Administrator"
password="password" />
```



3. Edit `DarInstall.bat` (Windows) or `DarInstall.sh` (Linux).

- a. Specify the path to the `composerheadless` package as the value of `ECLIPSE`.  
For example:

```
set ECLIPSE="C:\Documentum\product\6.6\install\composer\ComposerHeadless"
```

- b. Specify the path to the file DarInstall.xml in a temporary working directory (excluding the file name) as the value of *BUILDFILE*. For example:

```
set BUILDFILE="C:\DarInstall\temp"
```

- c. Specify a workspace directory for the generated Composer files. For example:

 **Example 10-6:**

```
set WORKSPACE="C:\DarInstall\work"
```



4. Launch DarInstall.bat (Windows) or DarInstall.sh (Linux) to install the query subscription SBO. On Windows 2008, run the script as administrator.

### 10.5.3 Testing query subscriptions

Before starting testing query subscriptions, make sure that the following prerequisites are satisfied:

- With Documentum Server 6.7 or lower, you installed the query subscription DAR file. See “[Installing the query subscription DAR](#)” on page 285.
- You installed JDK 7 or higher.
- With Documentum Server 7.0 or later, you activated query subscription jobs.
- You included the following JAR files in the Java classpath:

```
qbsAdmin.jar  
qbs-sbo.jar  
qbsTest.jar  
dfc.jar  
commons-lang-2.6.jar  
log4j.jar  
aspectjrt.jar
```

You execute the com.documentum.test.qbs.Tqbs.java program to test:

- Subscribing to a query for a specific user (-SubscribeAndVerify flag)
- Unsubscribing from a query for a specific user (-UnsubscribeAndVerify flag)
- Running a job and verifying that it completes successfully (-RunAndVerifyJob flag)
- Deleting a smartlist and verifying that the associated dm\_relation and dm\_ftquery\_subscription objects are deleted (-VerifyDeleteCascading flag)



**Note:** You can also use qbsadmin.bat or qbsadmin.sh. See the [qbsadmin.bat](#) and [qbsadmin.sh usage instructions](#).

1. If you have not done so already, create a `dm_smart_list` object, which is a saved query.

You can use a Documentum client (such as Webtop) to save a search, which creates a `dm_smart_list` object.

2. Execute the `Tqbs.java` class.

For example, executing the following command with the `-h` flag provides the syntax:

```
%JAVA_HOME%\bin\java" -classpath "C:\Documentum\config;C:\Documentum\Shared\dfc.jar;C:\Documentum\Shared\log4j.jar;C:\Documentum\Shared\commons-lang-2.6.jar;.\lib\qbsAdmin.jar;.\lib\qbs-sbo.jar;.\lib\qbsTest.jar"
com.documentum.test.qbs.Tqbs -h
```

## 10.5.4 Subscription reports

When you support query subscriptions, monitor the usage and query characteristics of the users with subscription reports. If there are many frequent or poorly performing subscriptions, increase capacity.

### Finding frequent or slow subscription queries

You can run the following reports to troubleshoot query subscription activity. Use the report to find both frequent and poorly performing queries.

- QBS activity report by user: Find the users whose subscription queries consume the most resources or perform poorly. Filter by date range. Set the number of users to display
- QBS activity report by ID: Find the subscription queries that consume the most resources or perform poorly. Filter by date range and user name. Order by Total processing time (descending), or Frequency (ascending). Set the number of IDs ( $N$ ) to display. If you order by total processing time,  $N$  subscriptions with the longest query times are displayed. If you order by job frequency,  $N$  subscriptions with the shortest job frequency are displayed.
- The Top N slowest query: Find the slowest subscription queries. Filter by *Subscription query* type.

### Finding subscriptions that use too many resources

Subscribed queries can consume too many resources. Use the *QBS activity report by ID* to find poorly performing queries. Use the *QBS activity report by user* to find users who consume the most resources. A user can consume too many resources with the following subscriptions:

- Too many subscriptions
- Queries that are unspecific (too many results)
- Queries that perform slowly (too many criteria)

## Users complain that subscriptions do not return results

Troubleshooting steps:

- Use Documentum Administrator to make sure that the job was run. Select the job report.
- Check that the user query was run using *QBS activity report by user*. If the query returns no results, set a lower frequency, depending on business needs.
- Check the query itself to see if it is properly formulated to return the desired results. The query may be searching on the wrong terms or attributes. In this case, reformulate the saved query.

### 10.5.5 Subscription logging

Subscribed queries are logged in dsearch.log with the event name QUERY\_AUTO. The following information is logged:

```
<event name="QUERY_AUTO" component="search" timestamp="2011-08-23T14:45:09-0700">
...
<application_context>
  <query_type>QUERY_AUTO</query_type>
  <app_name>QBS</app_name>
  <app_data>
    <attr name="subscriptionID" value="0800020080009561" />
    <attr name="frequency" value="DAILY" />
    <attr name="range" value="1015" />
    <attr name="jobintervalinseconds" value="86400" />
  </app_data>
</application_context>
</event>
```

Key:

- *subscriptionID* is set by the QBS application
- *frequency* is the subscription frequency as set by the client. Values: HOURLY, DAILY, WEEKLY, MONTHLY.
- *range* reports time elapsed since last query execution. For example, if the job runs hourly but the frequency was set to 20 minutes, the range is between 0 and 40 minutes (2400 seconds). Not recorded if the frequency is greater than one day.
- *jobintervalinseconds* is how often the subscription is set to run, in seconds. For example, a value 86400 indicates a setting of one day in the client. Not recorded if the frequency is greater than one day.

## 10.5.6 dm\_ftquery\_subscription

Represents a subscribed query.

### Description

Supertype: SysObject

Subtypes: None

Internal name: dm\_ftquery\_subscription

Object type tag: 08

A dm\_ftquery\_subscription object represents subscription-specific information but not the saved query itself, which is contained in a dm\_smart\_list object.

### Properties

The table describes the object properties.

**Table 10-5: dm\_ftquery\_subscription type properties**

Property	Datatype	Single or repeating	Description
frequency	CHAR(32)	S	How often the subscription is to run. Valid value is represented in frequency query subscription job parameter.
last_exec_date	TIME	S	Last date and time that the subscription was executed.
subscriber_name	CHAR(32)	S	Name of the user who uses this subscription.

Property	Datatype	Single or repeating	Description
zone_value	INTEGER	S	Zone to which this subscription belongs. With this value, all subscriptions with the same frequency can be picked by different jobs (User can customize jobs such that those jobs will be run on the same interval but with different value in job argument of "zone_value". Specify this value to run jobs when there are too many subscriptions for a single job.

Property	Datatype	Single or repeating	Description
result_strategy	INTEGER	s	<p>Integer that indicates whether existing results that are saved in the dm_smart_list are to be replaced with the new results (0, the default), merged with the new results (1), or the new results are to be discarded (2).</p> <p>The saved query results are sorted as follows:</p> <ul style="list-style-type: none"> <li>• If the result_strategy is set to 0, then the new results are sorted from the highest to lowest score.</li> <li>• If the result_strategy is set to 1, then the new results are listed first and sorted from the highest to lowest score; then the existing results are listed next and sorted from the highest to lowest score.</li> </ul>
workflow_id	ID	S	Process ID of the workflow to be executed by the job. If this value is null, then the notification is executed through queue item creation when any result is returned.

Property	Datatype	Single or repeating	Description
oldest_queueitem_date	TIME	S	Records the oldest queue item date_sent value when the subscription is run by the QBS job. When the job runs the same subscription the next time, it picks the value in the oldest_queueitem_date attribute if it is not NULL to cover the index latency issue.

### 10.5.7 dm\_qbs\_relation object

A dm\_relation\_type object that relates the subscription (dm\_ftquery\_subscription object) to the original dm\_smart\_list object.

**Table 10-6: dm\_qbs\_relation properties**

Property	Value	Notes
object_name	dm_qbs_relation	All query subscription-created dm_relation objects have this name.
parent_type	dm_smart_list	None.
child_type	dm_ftquery_subscription	None.
security_type	CHILD	None.
direction_kind	0	If a dm_smart_list object is deleted, then the corresponding dm_relation and dm_ftquery subscription objects are deleted.
integrity_kind	2	

## 10.5.8 Query subscription jobs

A job is executed for each query subscription at a specified interval. Results are returned via an inbox queue item and an email or a workflow. Query subscription jobs are inactive by default, if not already done, enable them in Documentum Administrator.

### Overview

Each of these jobs execute all query subscriptions that are specified to execute at the corresponding interval:

Job Name	Description
dm_FTQBS_HOURLY	Executes all query subscriptions that are to be executed once an hour.
dm_FTQBS_DAILY	Executes all query subscriptions that are to be executed once a day.
dm_FTQBS_WEEKLY	Executes all query subscriptions that are to be executed once a week.
dm_FTQBS_MONTHLY	Executes all query subscriptions that are to be executed once a month.

Each job executes its query subscriptions in ascending order based on each subscription last\_exec\_date property value. If a query subscription is not executed, it is executed when the job runs next.



**Note:** A job is stopped gracefully just before it is timed out.

### Method arguments

Argument	Description
-frequency	(Required) Selects the corresponding subscriptions. Valid values: hourly, daily, weekly, monthly
-stop_before_timeout	(Optional) Number of seconds before which you want the job to stop before timing out. Default value: 60.
-zone_value	(Optional) An integer that matches subscriptions with the same zone_value. If this argument is specified, then a dm_ftquery_subscription's zone_value and frequency attributes must match the corresponding method arguments in order for a subscription to be executed by the job.
-search_timeout	(Optional) Number of milliseconds that the job runs before it times out. Default: 60000.

Argument	Description
-max_results	(Optional) Maximum number of query results that can be returned as well as maximum number that can be saved in the subscription object. Default: 50.
-queueperson	(Optional) Fulltext index user. The default value is set to dm_fulltext_index_user. The value will not change unless the queue user of xPlore IA or the repository is different from the dm_fulltext_index_user and has different registered events for different types. For example, you will change the user in a dual mode setup.
-index_latency_in_secs	(Optional) If you know index latency between the time when the object is created and the object is indexed, you can add this argument so that the QBS job will not execute the DQL to get the oldest queue item and will always look back a specified number of seconds (in index_latency_in_secs) from last_exec_date when the QBS job runs the subscription next time.
-verbose	(Optional) The QBS job report will print more information for debugging purposes when it is turned on. The valid value is "T" or "t".
-qbsadmin	(Optional) Specify a valid user so that when job overloading occurs, the specified user receives a notification. When the QBS job is unable to finish all subscriptions because it is overloaded, the job sends a notification to the admin user. The default admin user who starts the QBS job is the installation owner. However, if a specific QBS admin user is set in the repository, you can have the QBS job send notifications to that user by enabling the "-qbsadmin" parameter.
-return_job_code	(Optional) Set the value to T to return the job code of the query execution with errors caught by QBS. When QBS jobs are running without errors, 0 is returned. The default value is false.
-repo_names	(Optional) A list of repositories on which the subscribed query runs separated by comma.
-repo_owners	(Optional) A list of repository owners of the repositories on which the subscribed query runs separated by comma.

## Reports

Job reports are stored in:

<DOCUMENTUM\_HOME>/dba/log/<sessionID/>sysadmin

Job Name	Report File
dm_FTQBS_HOURLY	FTQBS_HOURLYDoc.txt
dm_FTQBS_DAILY	FTQBS_DAILYDoc.txt
dm_FTQBS_WEEKLY	FTQBS_WEEKLYDoc.txt
dm_FTQBS_MONTHLY	FTQBS_MONTHLYDoc.txt

## Custom jobs

The job method `-zone_value` parameter is meant for partitioning the execution of query subscriptions among multiple custom jobs that run on the same interval. A custom job executes every `dm_ftquery_subscription` that has the same `zone_value` and `frequency` attribute values as the custom job. You must specify a `-zone_value` value for every custom job that runs on the same interval and that value must be unique amongst all those custom jobs. If a job does not specify a `-zone_value` value, then it will execute all subscriptions on the same interval regardless of each subscription's `zone_value` value.



**Note:** None of your custom jobs should have the same interval as any of the pre-installed jobs, because the pre-installed jobs do not have a `-zone_value` specified and will execute all subscriptions on the same interval regardless of their `zone_value` value.

### 10.5.9 Query subscription workflows

A workflow can be called by the query subscription job. Make sure that your workflow executes correctly before using it with query subscription jobs. For example, execution of the workflow will fail if the subscriber does not have at least RELATE permissions on the workflow.



**Note:** You can create a workflow using Documentum Process Builder.

## Requirements

- Activities:
  - One starting activity is required.
  - Only one starting activity can be specified.
  - The starting activity's name must be: QBS-Activity-1
- Packages:

- One package is required.
- Only one package can be specified.
- The package name must be: QBS-Package0
- The package type must be dm\_ftquery\_subscription.
- The subscription ID must be passed as the package.

## 10.5.10 IQuerySubscriptionSBO

Provides the functionality to subscribe to, unsubscribe from, and list query subscriptions.

### Interface name

com.documentum.server.impl.fulltext.qbs.IQuerySubscriptionSBO

### Imports

```
import com.documentum.server.impl.fulltext.qbs.IQuerySubscriptionSBO;
import com.documentum.server.impl.fulltext.qbs.QuerySubscriptionInfo;
import com.documentum.server.impl.fulltext.qbs.impl.QuerySubscriptionException;
```

### DAR

QBS.dar

### Methods

- public IDfId subscribe (String docbaseName, IDfId smartListID, String subscriber, String frequency, IDfId workFlowID, int zoneValue, IDfTime lastExecDate, int resultStrategy) throws DfException, QuerySubscriptionException

Validates the dm\_smart\_list object ID and subscriber name in the specified repository; validates the frequency value with all query

subscription jobs with the job method argument “-frequency”. Creates a dm\_ftquery\_subscription and dm\_relation objects. The object ID of dm\_ftquery\_subscription object is returned.

The workflow template ID can be set to null, if not applicable.

For zone\_value, specify -1, if not applicable.

For lastExecDate, specify DfTime.DF\_NULLDATE, if not applicable.

For resultStrategy: Integer that indicates whether existing results that are saved in the dm\_smart\_list are replaced with the new results (0, the default), merged with the new results (1), or the new results are discarded (2). Specify -1, if not applicable.

- public IDfId subscribe (String docbaseName, IDfId smartListID, String subscriber, String frequency, IDfId workFlowID, int zoneValue, IDfTime

```
lastExecDate, int resultStrategy, String subTypeName, Map  
customAttrAndValue) throws DfException, QuerySubscriptionException
```

You can create a subtype of `dm_ftquery_subscription` that has custom attributes. It enables you to display additional information related to the subscriptions in your application.

Creates a subscription with a subtype of `dm_ftquery_subscription` and its relation object based on the passed-in parameters.

The method parameters are similar to the ones of the previous method with two additional parameters: `subTypeName` and `customAttrAndValue`.

For `subTypeName`, specify the type name which is a subtype of `dm_ftquery_subscription`.

For `customAttrAndValue`, specify a map with attribute name and attribute value as key-value pair. For single-value attributes, indicate the value in its original datatype in value. For repeating attributes, indicate a List of values.

- `public boolean unsubscribe (String docbaseName, IDfId smartListID, String subscriber) throws DfException, QuerySubscriptionException`

Unsubscribe service destroys the `dm_relation` and `dm_ftquery_subscription` objects that are associated with the specified `dm_smart_list` and `subscriber`.

- `public List getSubscribedSmartList(String docbaseName, String subscriber) throws DfException`

Returns a list of all of the specified user subscriptions.

- `public QuerySubscriptionInfo getSubscriptionInfo(String docbaseName, IDfId smartlistId, String subscriber) throws DfException`

Returns information for a subscription based on the `dm_smart_list` object ID and `subscriber` name. The information includes: the `dm_smart_list` object ID and name, the subscription ID, the frequency, the workflow ID, the last execution date, and the zone value.

## For more information

- For more information about invoking SBOs, see the *OpenText Documentum Foundation Classes Development Guide*.
- For examples of calling this SBO, see the source code for the following class:
  - `com.documentum.test.qbs.Tqbs`
  - `com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool`
- To help you extend the functionality by creating subtypes of `dm_ftquery_subscription`, you can use the reference project `qbs.zip` located at `<xplore_home>/setup/qbs`. The *OpenText Documentum Composer 6.7 User Guide* describes how to designate a reference project. Use Composer 6.7 SP1 to load the query-based subscriptions reference project.

### 10.5.11 IQuerySubscriptionTBO

Manages basic query subscription execution.

#### Interface name

com.documentum.server.impl.fulltext.qbs.IQuerySubscriptionTBO

#### Imports

```
import com.documentum.server.impl.fulltext.qbs.IQuerySubscriptionTBO;
import com.documentum.server.impl.fulltext.qbs.results.DfResultsSetSAXDeserializer;
```

#### DAR

QBS.DAR

#### Methods

- `public void setSmartListId(IDfId smartListId)`  
Sets the dm\_smart\_list object ID associated with the dm\_ftquery\_subscription object. This method must be called before calling runRangeQuery().
- `public IDfResultsSet runRangeQuery(String docbaseName, IDfTime from) throws DfException, IOException, InterruptedException`  
Executes a query saved in a dm\_smart\_list object from the specified date/time in the `from` parameter. If `from` is not a nulldate, a range is added to the search query with a condition like “r\_modify\_date >= from”. If `from` is a nulldate, then no range condition is added to the search query.
- `public void setResults(IDfResultsSet results)`  
Saves the results to dm\_ftquery\_subscription.
- `public IDfResultsSet getResults() throws DfException`  
Gets the results that are saved in dm\_ftquery\_subscription.
- `public void setSearchTimeOut(long timeout)`  
Sets the number of milliseconds that the search runs before it times out.
- `public long getSearchTimeOut()`  
Gets the number of milliseconds that the search runs before it times out.
- `public void setMaxResult(int max); public int getMaxResult()`  
Sets the maximum number of query results that can be returned as well as maximum number that can be saved in the subscription object.
- `public int getMaxResult()`  
Gets the maximum number of query results that can be returned as well as maximum number that can be saved in the subscription object.
- `public void setResultStrategy(int resultStrategy)`

Integer indicates whether existing results that are saved in the dm\_smart\_list are replaced with the new results (0, the default), merged with the new results (1), or the new results are discarded (2).



**Note:** doSave() updates the last\_exec\_date of the subscription based on this value.

- `public int getResultStrategy()`  
Gets the result strategy.
- `public void setQBSAttrs(Map<String, String>qbsInfo)`  
Sets query subscription information, such as subscription ID.
- `public Map<String, String> getQBSAttrs()`  
Returns a hash map contains key-value pairs for query subscription information.

## Notes

Extending this TBO is not supported.

## For more information

- See the Javadoc for more information about this TBO.
- For more information about invoking TBOs, see the *OpenText Documentum Foundation Classes Development Guide*.

## 10.5.12 QuerySubscriptionAdminTool

### Class name

`com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool`

### Usage

You use `com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool` to:

- Subscribe to a query for a specific user (-subscribe flag)
- Unsubscribe from a query for a specific user (-unsubscribe flag)
- List all subscribed queries for a specific user (-listsubscription flag)



**Note:** All parameter values are passed as string values and must be enclosed in double quotes if spaces are specified in the value.

To display the syntax, specify the -h flag. For example:

```
C:\Temp\qbsadmin>"%JAVA_HOME%\bin\java" -classpath "C:\Documentum\config;C:\Documentum\Shared\aspectjrt.jar;C:\Documentum\Shared\dfc.jar;C:\Documentum\Shared\log4j.jar;C:\Documentum\Shared\commons-lang-2.6.jar;..\lib\qbsAdmin.jar;..\lib\qbs-sbo.jar;..\lib\qbsTest.jar" com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool -h
```



**Note:** In <xplore\_home>/setup/qbs/tool, qbsadmin.bat and qbsadmin.sh demonstrate how to call this class. In qbsadmin.bat and qbsadmin.sh, modify the path to the dfc.properties file. You can also change the -h flag to one of the other flags.

## Required JARs

qbs-sbo.jar

qbsAdmin.jar

qbsTest.jar

dfc.jar

log4j.jar

commons-lang-2.6.jar

aspectjrt.jar

### ➤ Example 10-7: -subscribe example

```
C:\Temp\qbsadmin>"%JAVA_HOME%\bin\java" -classpath "C:\Documentum\config;C:\Documentum\Shared\aspectjrt.jar;C:\Documentum\Shared\dfc.jar;C:\Documentum\Shared\log4j.jar;C:\Documentum\Shared\commons-lang-2.6.jar;.\lib\qbsAdmin.jar;.\lib\qbs-sbo.jar;.\lib\qbsTest.jar" com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool-subscribe SAMPLEDOCBASE user1 password password1 080000f28002ef2c daily
```



### ➤ Example 10-8: -subscribe output

```
subscribed 080000f28002ef2c for user user1 succeeded with subscription id 080000f28002f115
```



### ➤ Example 10-9: -unsubscribe example

```
C:\Temp\qbsadmin>"%JAVA_HOME%\bin\java" -classpath "C:\Documentum\config;C:\Documentum\Shared\aspectjrt.jar;C:\Documentum\Shared\dfc.jar;C:\Documentum\Shared\log4j.jar;C:\Documentum\Shared\commons-lang-2.6.jar;.\lib\qbsAdmin.jar;.\lib\qbs-sbo.jar;.\lib\qbsTest.jar" com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool-unsubscribe SAMPLEDOCBASE user1 password passwrod1 080000f28002ef2c
```



### ➤ Example 10-10: -unsubscribe output

```
User user1 has no subscriptions on dm_smart_list object (080000f28002ef2c)
```



▶ **Example 10-11: -listsubscription example**

```
C:\Temp\qbsadmin> "%JAVA_HOME%\bin\java" -classpath "C:\Documentum\config;C:\Documentum\Shared\aspectjrt.jar;C:\Documentum\Shared\dfc.jar;C:\Documentum\Shared\log4j.jar;C:\Documentum\Shared\commons-lang-2.6.jar;.\lib\qbsAdmin.jar;.\lib\qbs-sbo.jar;.\lib\qbsTest.jar" com.documentum.server.impl.fulltext.qbs.admin.QuerySubscriptionAdminTool-listsubscription SAMPLEDOCBASE user1 password password1
```



▶ **Example 10-12: -listsubscription output**

```
Subscriptions for user1 are:  
smartList: 080000f28002ef2c frequency: DAILYworkFlowID: 0000000000000000  
smartList: 080000f28002ef2f frequency: 5 MINUTESworkFlowID: 0000000000000000
```



## 10.6 Troubleshooting search

When you set the search service log level to WARN, queries are logged. “[Auditing queries](#)” on page 302 describes how to view or customize reports on queries.

### Testing a query in xPlore Administrator

You can perform a test search on a full-text string in content or metadata:

1. In xPlore Administrator, expand the **Diagnostic and Utilities** tree and choose **Test search**.
2. To search for a keyword, choose **Keyword** and enter the search string. If you enter multiple terms in the Keyword field, the XQuery expression is generated using the AND condition.
3. To search using an XQuery expression, choose **XQuery** and enter the expression. Make sure that you select the correct domain.

The results of the query along with the **Execute** and **Fetch** times are displayed. In addition, the button **View audit record** is displayed above the result. Clicking this button displays the audit record for the query.

By default, a super user executes the query, so security is not evaluated. If you want to simulate a normal user, click **Add options**, then select **User** to specify the user name. Select **Timeout(ms)** and specify a timeout value. The search results will be filtered.



**Note:** If the timeout(ms) option of the test search is not set, it will use the maximum value of 10 minutes and the query-default-timeout setting configured in indexserverconfig.xml.

### Testing a query in Documentum iAPI or DQL

Try a query like the following:

```
api>?,c,SELECT text,object_name FROM dm_document SEARCH DOCUMENT CONTAINS 'test'  
WHERE (a_is_hidden = FALSE)
```

## Debugging from Webtop

If the query fails to return expected results in Webtop, perform a Ctrl-click on the **Edit** button in the results page. The query is displayed in the events history as a select statement like the following:

```
IDfQueryEvent(INTERNAL, DEFAULT): [dm_notes] returned [Start processing] at  
[2010-06-30 02:31:00:176 -0700]  
IDfQueryEvent(INTERNAL, NATIVEQUERY): [dm_notes] returned  
[SELECT text,object_name,score,summary,r_modify_date,...  
SEARCH DOCUMENT CONTAINS 'ctrl-click' WHERE (...)]
```

If there is a processing error, the stack trace is shown.

## Debugging from DFC

To log XQuery and XML results, set `log4j.logger.com.documentum.fc.client.search=DEBUG`, `stdout` in `dfc.properties` for the DFC application. The file `dfc.properties` is located in the WEB-INF/classes directory of a web application like Webtop or CenterStage.

### Determining the area of failure

1. If the query runs successfully in xDB, use xPlore administrator to run the XQuery (**Execute XQuery** in the domain or collection view).
2. If xPlore administrator runs the query successfully, check the query plugin trace log. See “[Tracing Documentum queries](#)” on page 280.
3. If there are two `counter.xml` files in `domain_name/dsearch/ApplicationInfo/group`, delete the file that contains the lower integer value.

### 10.6.1 Auditing queries

Auditing is enabled by default. Audit records are purged on a configurable schedule (default: 30 days).

To enable or disable query auditing, open **System Overview** in the xPlore administrator left pane. Click **Global Configuration** and choose the **Auditing** tab. Click **search** to enable query auditing.

For information on configuring the audit record, see “[Configuring the audit record](#)” on page 48.

Audit records are saved in an xDB collection named `AuditDB`. You view or create reports on the audit record. Query auditing provides the following information:

- The XQuery expression in a CDATA element.
- The user name and whether the user is a superuser.

- The application context, in an application\_context element. The application context is supplied by the search client application.
- The query options in name/value pairs set by the client application, in the QUERY\_OPTION element.
- The instance that processed the query, in the NODE\_NAME element.
- The xDB library in which the query was executed, in the LIBRARY\_PATH element.
- The number of hits, in the FETCH\_COUNT element.
- The number of items returned, in the TOTAL\_HITS element.
- The amount of time in msec to execute the query, in the EXEC\_TIME element.
- The time in msec elapsed to fetch results, in the FETCH\_TIME element.
- The following security events are recorded for user-generated queries. The audit record reports how many times these caches were hit for a query. For details on configuring the caches, see “[Configuring the security cache](#)” on page 67.
  - How many times the group-in cache was probed for a query, in the GROUP\_IN\_CACHE\_HIT.
  - GROUP\_IN\_CACHE\_FILL - How many times the query added a group to the group-in cache.
  - TOTAL\_INPUT\_HITS\_TO\_FILTER - How many hits a query had before security filtering.
  - Number of hits filtered out by security because the user did not have sufficient permission, in the HITS\_FILTERED\_OUT element.
  - Number of times the owner group cache is filled, in the OWNER\_GROUP\_CACHE\_FILL element.
- The status of the query, in the STATUS element.

To view a query in the audit record, go to **Diagnostic and Utilities > Reports** and choose **Audit records for search component**. You can filter by date and query type. From the results, click a query of interest to view the XML entry in the report. The XQuery expression is contained within the QUERY element.

To view warmup queries in the audit record, choose the query type *Warmup ToolQuery* in the *Audit records for search component* report..

## 10.6.2 Search is not available

### Full-text service is disabled

Investigate the following possible causes:

- No queries are allowed when the search service has not started. You see this error message:  

```
The search has failed: The Full-text service is disabled
```
- The username contains an illegal character for the xPlore host code page.
- The wrong query plugin is in use. See “[Query plugin configuration \(dm\\_ftengine\\_config\)](#)” on page 273.

### Verifying the query plugin version

Queries fail if the wrong (FAST) query plugin is loaded in the Documentum Server. Check the Documentum Server log after you start the Documentum Server. The file *repository\_name.log* is located in <DOCUMENTUM\_HOME>/dba/log. Look for the line like the following. It references a plugin with DSEARCH in the name.

```
[DM_FULLTEXT_T_QUERY_PLUGIN_VERSION]info: "Loaded FT Query Plugin:  
.../DSEARCHQueryPlugin.dll...FT Engine version: X-Hive/DB 10"
```

The Documentum Server query plugin properties of the *dm\_ftengine\_config* object are set during xPlore configuration. If you have changed one of the properties, like the primary xPlore host, the plugin can fail. Verify the plugin properties, especially the qrserverhost, with the following DQL:

```
1> select param_name, param_value from dm_ftengine_config  
2> go
```

### Connection refused or no collection available

If an API returns a connection refused error, check the value of the URL on the instance. Make sure that it is valid and that search is turned on for the instance. If the search service is not enabled, dsearch.log records the following exception:

```
com.emc.documentum.core.fulltext.common.search.FtSearchException:...  
There is no node available to process this request type.
```

From Documentum DFC clients, the following exception is returned:

```
DfException: ..."EXEC_XQUERY failed with error:  
ESS_DMSearch:ExecuteSearchPassthrough. Communication Error  
Could not get node information using round-robin routing.
```

From Documentum DQL, the following error is returned:

```
dmFTSearchNew failed with error:  
ESS_DMSearch:ExecuteSearch. Communication Error  
Could not get node information using round-robin routing.
```

### Error after you changed xPlore host

If you have to change the xPlore host, do the following:

- Update indexserverconfig.xml with the new value of the URL attribute on the `node` element. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.
- Change the WildFly startup (script or service) so that it starts correctly.

### 10.6.3 Troubleshooting slow queries

Check the audited search events using xPlore administrator.

#### The attribute cannot be searched

All attributes can be found in a full-text search, but this search can be very slow or time out. Attributes that are commonly searched should be defined in a subpath definition in indexserverconfig.xml.

Sometimes the datatype is incorrectly specified in indexserverconfig.xml. By default, a custom attribute is treated as a string datatype. Boolean or datetime attributes are not returned. If the custom attribute is not a string, set the appropriate type in the subpath definition. Valid values: string | integer | double | date | datetime. Boolean is not supported.

After you change a subpath, you must reindex. If the attribute is for a new custom type, reindexing is not required.

#### The query does not use the index

For slow queries, always check whether the query is running as NOFTDQL against the database (not xPlore). If a multi-path index is not used to service the query, then the query runs slowly. DQL and DFC Search service queries always use the index unless there is a NOFTDQL hint. Some IDfXQuery-based queries might not use the index.

To detect this issue with query auditing, find the query using the TopNSlowestQueries report (with user name and day). Click the query ID to get the query text in XML format. Obtain the query plan to determine which indexes were probed, if any. (Provide the query plan to OpenText Global Technical Services for evaluation.) Rewrite the query to use the index.

*Test the query in the xDB admin tool*

Test the query in xDB admin and see if “Using query plan: Index(dmftdoc)/child::dmftkey” is in the query debug output. If not, the query is NOFTDQL (database).

To detect queries that do not use the index (NOFTDQL queries), turn on full-text tracing in the Documentum Server:

```
API>apply,c,NULL,MODIFY_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,all
```

Look for temp table creation and inserts like the following:

```
Thu Feb 09 15:50:12 2012 790000: 6820[7756] 0100019f80023909
process_ftquery_to_temp --- will populate temp table in batch size 20000
    Thu Feb 09 15:50:12 2012 790000: 6820[7756] 0100019f80023909
build_fulltext_temp --- begin: create the fulltext temporary table.
    Thu Feb 09 15:50:13 2012 227000: 6820[7756] 0100019f80023909
BuildTempTbl --- temporary table dmft80023909004 was created successfully.
    Thu Feb 09 15:50:13 2012 430000: 6820[7756] 0100019f80023909
Inserting row at index 0 into the table
```

## Security caches are not tuned

If a user is very underprivileged, or the user is the member of many groups, queries can slow due to small group caches. For instructions on configuring the caches, see “[Configuring the security cache](#)” on page 67.

## Result sets are large

By default, xPlore gets the top 12,000 most relevant results per collection to support a facet window of 10,000 results. Webtop applications consume only 350 results, so the extra result memory is costly for large user environments or multiple collections (multiple repositories). In an environment with millions of documents and multiple collections, you could see longer response times or out of memory messages.

Custom clients can consume a larger result set. Make sure that query auditing is enabled (default). Examine the number of results in the TopNSlowestQueries report for a specific user and day. If the number of results is more than 1000, the custom client might return all the results.

Workarounds:

- Limit query result set size: Open xdb.properties, which is located in the directory WEB-INF/classes of the primary instance. Set the value of queryResultsWindowSize to a number smaller than 12000.
- Change the client to consume a smaller number of results by closing the result collection early or by using the DQL hint ENABLE(RETURN\_TOP\_N).

## xPlore security is disabled

When xPlore native security is disabled, Documentum Server security is much slower than xPlore native security, because some or many results that are passed to the Documentum Server are discarded. To detect the problem with query auditing enabled, examine the number of results in the TopNSlowestQueries report for a specific user and day. If the number of results is more than 1000, xPlore security might be disabled and the user is underprivileged. (When the user is underprivileged, many results are discarded.)

*Workaround:* Enable xPlore native security. See “[Changing search results security](#)” on page 63.

## FAST-compatible wildcard and fragment behavior is enabled

Many Documentum clients do not enable wildcard searches for word fragments like “car” for “careful.” The FAST indexing server supported word fragment searches for leading and trailing wild cards in metadata. If you enable FAST-compatible wildcard behavior for your Documentum application, you see slower queries when the query contains a wildcard.

You can fine tune your wildcard support with settings for content or metadata, implicit, or explicit..

## Slow queries due to frequent document updates

If the update rate is high and the query rate is low, queries can be impacted. Add the following property to xdb.properties, which is located in *xplore\_home/ <wildfly\_version>/server/instance\_name/deployments/dsearch.war/WEB-INF/classes*.

```
xdb.lucene.refreshBlacklistCacheDuringMerge
```

This property value is false by default. When set to true, blacklist caches are refreshed during non-final merges.

## Insufficient CPU, disk I/O, or memory

If a query is slow the first time and much faster when it is reissued, the problem is likely due to insufficient I/O capacity on the allocated drives. Concatenated drives often show much lower I/O capacity than striped drives because only a subset of drives can service I/O requests.

Workarounds:

- If the system has only one or two cores and a high query rate, add more CPUs.
- If the system is large but receives complex or unselective queries, make sure that query auditing is enabled. Examine the TopNSlowestQueries report for the specific user name and day on which the problem was experienced. Look for high query rates with slow queries. Add more capacity.

## Query of too many collections

A query probes each index for a repository (domain) sequentially. Results are collected across repositories. If there are multiple repositories, or multiple collections within a domain, the query can take more time. With query auditing enabled, try the query across repositories and then target it to a specific repository.

Use one of the following solutions for this problem:

- Merge collections using xPlore administrator. See “[Moving a collection](#)” on page 202.
- Use parallel queries in DFC-based search applications by setting the following property to true in dfc.properties:

```
dfc.search.xquery.option.parallel_execution.enable = true
```

- Use the ENABLE(fds\_collection collectionname) hint or the IN COLLECTION clause in DQL. See “[Routing a query to a specific collection](#)” on page 317

### User is very underprivileged

If the user is very underprivileged, the security filter discards tens of thousands of results. To detect this problem with query auditing, find the query using the TopNSlowestQueries report for the specific user and day and check if the number in the Documents filtered out columns is large.

*Workaround:* Queries can generally be made more selective. If you cannot modify the query, organize the repository so that the user has access to documents in certain containers such as rooms or cases. Append the container IDs to the user query. Enlarging the security cache size can improve the performance. See “[Configuring the security cache](#)” on page 67.

### Non-string metadata searches are too slow

By default, all metadata is indexed as string type. Add a subpath definition to indexserverconfig.xml for non-string metadata. To speed up searches for non-string attributes, add a subpath like the following. Valid values: string | integer | boolean | double | date | datetime.

```
<sub-path path="dmftmetadata//r_creation_date" type="datetime"
full-text-search="false" value-comparison="true"/>
```



**Note:** If the metadata is used to compute facets, set returning-contents to true.

## 10.6.4 Unexpected search results

### The wrong number of results are returned

There is latency between document creation or modification and indexing. First, check whether the object has been indexed yet. You can use the following DQL. Substitute the actual object ID of the document that exists on the Documentum Server but is not found in search results:

```
select r_object_id from dm_sysobject search document contains object_id
```

If the object has been indexed, check the following:

- Check user permissions. Run the query as superuser or through xPlore administrator.
- ACL and group can be out of synch between repositories and xPlore. Run the manual update script *acreplication*. See “[Manually updating security](#)” on page 64.
- Query tokens might not match indexed tokens (because of contextual differences). Run the tokenization test on the query terms and on the sentence containing the terms in the document. See “[Troubleshooting content processing](#)” on page 143

- Make sure that the attribute was not excluded from tokenization. Check indexserverconfig.xml for a subpath whose full-text-search attribute is set to false, for example:

```
<sub-path ...full-text-search="false" ...path="dmftmetadata//acl_name" />
```

- Make sure that counter.xml has not been deleted from the collection `domain_name/dsearch/ApplicationInfo/group`. If it has been deleted, restart xPlore.
- Try the query with Documentum Server security turned on. (See “[Changing search results security](#)” on page 63.)
- Summary can be blank if the summary security mode is set to BROWSE. (See “[Configuring summary security](#)” on page 263.)

## Changes to configuration are not seen

If you have edited indexserverconfig.xml, your changes are not applied unless the system is stopped. Some changes, such as adding or modifying a subpath do not take effect on a restart but only when you rebuild the indexes of the collection.

[“Modifying indexserverconfig.xml”](#) on page 54 describes the procedure to view and update the index configuration.

Try the following troubleshooting steps:

1. Make sure that the indexing status is DONE. See “[Troubleshooting indexing](#)” on page 182
2. Verify that the document was indexed to the correct collection. The collection for each document is recorded in the tracking DB for the domain. Substitute the document ID in the following XQuery expression, and execute it in the xDB admin tool:
 

```
for $i in collection("dsearch/SystemInfo")
where $i//trackinginfo/document[@id="TestCustomType_txt1276106246060"]
return $i//trackinginfo/document/collection-name
```
3. Set the `save-tokens` option to true for the target collection and restart xPlore, then reindex the document. Check the tokens in the Tokens library to see whether the search term was properly indexed.

## Document is not indexed

Objects of a certain type are not indexed. The query is not recorded in `dsearch.log`, and the query is executed in the database (case-sensitive). Check the properties for the object type in Documentum Administrator. Check **Enable Indexing: Register for indexing**. You can submit the specific type for indexing using the index agent UI.

## Search is case sensitive

If a query is not full-text compliant, or if the object type is not indexing-enabled in Documentum Administrator, the query is executed against the database. Database queries are case sensitive.

## Document is indexed but not searchable

Try the following troubleshooting steps:

- Make sure that the indexing status is DONE. (See “[Troubleshooting the index agent](#)” on page 111.)
- Verify that the document was indexed to the correct collection. The collection for each document is recorded in the tracking DB for the domain. Substitute the document ID in the following XQuery expression, and execute it in the xDB admin tool:

```
for $i in collection("dsearch/SystemInfo")
where $i//trackinginfo/document[@id="TestCustomType_txt1276106246060"]
return $i//trackinginfo/document/collection-name
```

- Set the *save-tokens* option to true for the target collection and restart xPlore, then reindex the document. Check the tokens in the Tokens library to see whether the search term was properly indexed.

## Foreign language is not identified

Queries issued from Documentum clients are searched in the language of the session\_locale. The search client can set the locale through DFC or iAPI. Because queries are shorter than indexed content, the language can be misidentified more easily than during indexing. To see the language that was identified during indexing, open the dftxml in xPlore administrator. For example:

```
<dmftdoc dmftkey="090012a78000e77a" dss_tokens=":dftxml:1" lang="it">
```

To see the language that was identified in the query, view the query in dsearch.log. For example:

```
<message>
<![CDATA[QueryID=primary$f20cc611-14bb-41e8-8b37-2a4f1e135c70,
query-locale=en,...>
```

Search results differ when searching with different locales, especially compound terms that have associated components. For example, a search for *Stollwerk* returned many more results when using the German than the English locale. *Stollwerk* is lemmatized as *stollwerk* in English but as *stoll* and *werk* in German. You can turn off lemmatization. See “[Configuring indexing lemmatization](#)” on page 133.

## Document is not found after index rebuild

At the original ingestion, some documents are not embedded in dftxml. In these documents, the XML content exceeds the value of the file-limit attribute on the xml-content element in indexserverconfig.xml. The index rebuild process generates a list of object IDs for these documents. The list is located in *xplore\_home/data/domain\_name/collection\_name/index\_name/ids.txt*, for example:

```
C:/xPlore/data/mydomain/default/dmftdoc_2er90/ids.txt
```

Reingest these large documents.

## Search for XML fails

Users can search for a specific element in an XML document. By default, XML structure of an input document is not indexed. To support search in XML content or attributes, change this setting in `indexserverconfig.xml` (For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54) and perform steps to add XML zone search support (see “[Supporting search in XML documents](#)” on page 247). If your documents containing XML have already been indexed, they must be reindexed to include the XML content.

- Change the value of the `store` attribute on the `xml-content` element to `embed`.
- Change the value of the `tokenize` attribute on the `xml-content` element to `true`.



**Note:** When the `store` attribute is set to `none`, XML documents are not tokenized even if the `tokenize` attribute is set to `true`.

- Change the value of the `index-as-sub-path` attribute on the `xml-content` element to `true`.
- Verify the `path` value attribute on the `xml-content` element against the `dftxml` path. (For the `dftxml` DTD, see “[Extensible Documentum DTD](#)” on page 413.) An XPath error can cause the query to fail.

## Wildcard searches fail

By default, wildcards in a search match words in the content of a document, not fragments of a word. (Wildcards are supported in attribute searches.) The FAST flag `fds_contain_fragment` parameter is not evaluated for xPlore.

You can turn on fragment support: Set the `fast_wildcard_compatible` parameter in the `dm_ftengine_config` object to `true`. Applies to both metadata and search document contains (simple one-box) search.

### 10.6.5 Debugging queries

You can debug queries for the following problems:

- Query does not return expected results.
- Query is very slow (reported in *Top N Slowest Queries* report).

## Debugging a query that does not return results

One of the most common causes of no results is a search for non-string metadata (integer or date) that is not in the index. The following example executes a query from the domain **Execute XQuery** command. If you input a value in Test Search, you can get the XQuery expression to provide to **Execute XQuery**.

```
for $i score $s in /dmftdoc[. ftcontains '9001001' with stemming]
order by $s descending return <d> {$i/dmftmetadata//r_object_id}
{ $i/dmftmetadata//object_name } { $i/dmftmetadata//r_modifier } </d>
```

No results are returned, because the searched value is a Documentum integer attribute. When you execute the query with the **get query debug** option, you see that the value is treated as a string:

```
query:1:20:for expression .../child::dmftdoc[. contains text 9001001]
```

Every library is visited, but no result is found:

```
query:1:20:for expression .../child::dmftdoc[. contains text 9001001]
query:1:20:No query plan with indexes found, walking descendants
query:1:67:No indexes found to support order specs
```

You must stop the xPlore instances and add a subpath for the non-string attribute. In this example, the following subpath was added to the dmftdoc category. Note that partial paths are supported, in case the metadata value is found in more than one path:

```
<sub-path description="award number" returning-contents="true"
value-comparison="true" type="integer" path="dmftmetadata//award_no"/>
```

For the non-string value to be found, we must reindex the domain (or the specific collection, if known). After reindexing, we have a different result in **Test Search**:

```
query:1:99:Using query plan:
query:1:99:index(dmftdoc)
query:1:99:Looking up "(false, true, 030012a7800001dc, 1001)" in index "dmftdoc"
query:1:290:Found an index to support all order specs. No sort required.
```

## Debugging a slow query

Get the XQuery from the *Top N Slowest Queries* report by clicking the query ID. You see the query in a window like the following:

**Figure 10-3: Query**

Open the Data Management tree and choose the library that contains your index. For Documentum environments, the library name is the same as the repository name. Click **Execute XQuery** and input the query from your text editor. Choose the option **Get query debug**. Click **Execute XQuery**.

### Getting the query execution plan

The query plan can be useful to OpenText Global Technical Services for evaluating slow queries. The query plan shows which indexes were probed and the order in which they were probed. Use one of the following options to save or fetch the query plan:

- *Using DFC query builder API*

save:

```
IDfxQuery.setBooleanOption(IDfxQuery.FtQueryOptions.
SAVE_EXECUTION_PLAN,true)
```

retrieve:

```
IDfxQuery.getExecutionPlan(session)
```

To get an execution plan, do the following:

```
xquery.setXQueryString(statement);
IDfxQueryTargets target = new DfFullTextXQueryTargets();
```

```

xquery.setIntegerOption(IDfXQuery.FtQueryOptions.BATCH_SIZE, 10);
xquery.setBooleanOption(IDfXQuery.FtQueryOptions.SAVE_EXECUTION_PLAN, true);
xquery.setBooleanOption(IDfXQuery.FtQueryOptions.CACHING, true);
xquery.setIntegerOption(IDfXQuery.FtQueryOptions.TIMEOUT, 600000);
xquery.setBooleanOption(IDfXQuery.FtQueryOptions.RETURN_TEXT, true);
xquery.setBooleanOption(IDfXQuery.FtQueryOptions.RETURN_SUMMARY, true);
xquery.setStringOption(IDfXQuery.FtQueryOptions.APPLICATION_NAME, "DfxQuery");

xquery.execute(session, target);
try
{
    String plan = xquery.getExecutionPlan(session);
    System.out.println("xquery plan = " + plan);
}
catch (DfException e)
{
    System.out.println(e.getMessage());
}

```

- Using iAPI

save:

```
apply,c,NULL,MODIFY_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,ftengine
```

retrieve: The query execution plan is written to `dsearch.log`, which is located in the logs subdirectory of the WildFly deployment directory.

- Using xPlore search API

save:

```
IDfXQuery.setSaveExecutionPlan(true)
```

retrieve:

```
IFtSearchSession.fetchExecutionPlan(requestId)
```

## Debugging Webtop queries

For queries from Webtop or other WDK applications, you can get the query by performing a Ctrl-click on the **Edit** button in the results page. A query window displays events history. The internal *nativequery* event contains the XQuery. (You must view the source to get the proper form of the XQuery, stripping out the enclosing brackets and timestamp.)

```

let $libs :=
('/TechPubsGlobal/dsearch/Data') let $results :=
for $dm_doc score $s in collection($libs)/dmftdoc[
(dmftmetadata//a_is_hidden = "false") and (
dmftversions/iscurrent = "true")
and (. ftcontains "award" with stemming)]
order by $s descending
return $dm_doc return (for $dm_doc in subsequence($results,1,351)
return <r>{for $attr in $dm_doc/dmftmetadata/*[local-name()=(
'object_name','r_modify_date','r_object_id','r_object_type',
'r_lock_owner','owner_name','r_link_cnt','r_is_virtual_doc',
'r_content_size','a_content_type','i_is_reference','r_assembled_from_id',
'r_has_fnr_assembly','a_compound_architecture','i_is_replica',
'r_policy_id')]
return <attr name='{local-name($attr)}' type='{$attr/@dmfttype}'>{string($attr)}</attr>}{xhive:highlight(($dm_doc/dmftcontents/
dmftcontent/dmftcontentref,$dm_doc/dmftcustom))}<attr name='score' type='dmdouble'>{string(dsearch:get-score($dm_doc))}</attr></r>
)

```



**Note:** The XQuery portion of the query is almost identical to the query retrieved through xPlore administrator. These queries were issued separately, which accounts for differences.

To debug the Webtop query, edit the query from View Source and enter it in the Execute XQuery dialog box in xPlore administrator.

## 10.6.6 About Testing Text Extraction

The **Test Text Extraction** page enables you to test the text extraction from a specific file. This can help you to diagnose problems or validate the resolution of issues with text extraction from a specific file.

The file from which you want to extract the text can be uploaded either from the local file system or from a remote path that is accessible from the selected CPS system. Plain text is extracted from the file and displayed within the **Test Text Extraction** page. In addition to the extracted content, the time taken to extract the text is also displayed.

The **Test Text Extraction** page allows you to select and use any of the available CPS instances for text extraction.

This feature supports extraction of text from several different file formats. The **Content Extractor** field allows you to select the content extractor module that supports the type of file from which you want to extract text. Typically, this field has two options: XML and Stellent. You can use the XML content extractor only for files with XML content. The Stellent content extractor can be used for XML, PDF, Microsoft Office, and other file formats.



**Note:** You can also use other content extractor modules such as Apache Tika. For more information about the content extractor modules, see [Custom Content Processing](#).

Similar to the regular text extraction operations in the CPS, the **Test Text Extraction** page is restricted by certain CPS configuration parameters.



**Note:** If you want to test large files, you should consider changing the relevant CPS parameters. For more information, see the section [Very large documents in Maximum document and text size](#). As changes to these CPS parameters might degrade xPlore performance, you should be careful while making the changes.

### 10.6.7 Testing Text Extraction

1. To expand the **Diagnostic and Utilities** menu, click the + sign next to **Diagnostic and Utilities**.
2. Click **Test Text Extraction**.
3. In the **Choose CPS** list, select the CPS node that you want to use.

 **Note:** You must select a local CPS instance if you want to test a file on the local file system. If you select a remote CPS instance, you must provide a remote file path as described in [Step 6](#).
4. In the **Content Extractor** list, select the content extractor module you want to use.
5.  **Note:** For an XML file, select **XML**. For other formats, select **Stellent**. For more information about supported file formats, see [Text extraction](#).
6. If you have selected the local CPS instance, select **Local File** and click **Choose File** to select the required file.
7. If you have selected a remote CPS instance, select **Remote File** and provide the complete path to the remote file.
7. Click **Extract Content**.

The **Test Results** box displays the text extracted from the file. The time taken to extract the content is displayed above the box.

### 10.6.8 Rerun a Search Query from an Audit Report

In some situations, you might want to check the response and the execution and fetch times of a search query that was run previously. For example, after making changes to an xPlore Index Agent or a CPS to resolve slow-running search queries, you might want to rerun the queries to verify your changes.

The Diagnostic and Utilities > Reports page enables you rerun a query directly from the report. This feature is available with the following reports:

- Get Query Text
  - Audit Records for Search Component
1. To expand the **Diagnostic and Utilities** menu, click the + sign next to **Diagnostic and Utilities**.
  2. Click **Reports**.
  3. In the **View** list, select either the desired report.



**Note:** Currently, only the **Get Query Text** and the **Audit Records for Search Component** reports support this feature.

4. Select the other desired options for the report and click **Run**.
5. In results table, click on a query in the **Query** column.  
A popup window displays the query in the XQuery format.
6. Click **Rerun Query**.
7. The **Test Search** page is displayed with the selected query copied to the XQuery text box.
8. Add the required search options, if any.
9. Click **Search**.  
The results of the query with the Execution and Fetch times are displayed.

## 10.7 Routing a query to a specific collection

You can route a query to a specific in the following ways:

- Route an individual query using the DQL *in collection* clause to specify the target of a SELECT statement. By default, DFC does not generate DQL, but you can turn off XQuery generation. See “[Turning off XQuery generation to support DQL](#)” on page 318.

For example:

```
? ,c,select r_object_id from dm_document search document contains 'benchmark'
in collection('custom')
```

- Route all queries that meet specific criteria using a DQL hint in dfcdqlhints.xml *enable(fds\_query\_collection\_collectionname)* where collectionname is the collection name. If you use a DQL hint, you do not need to change the application or DFC query builder. You must turn off XQuery generation. See “[Turning off XQuery generation to support DQL](#)” on page 318.) . For more information on the hints file, refer to *OpenText Documentum Search Development Guide*. For example:

```
? ,c,select r_object_id from dm_document search document contains 'benchmark'
enable(fds_query_collection_custom)
```

- Implement the DFC query builder API addPartitionScope.
- Implement the DFC IDfXQuery API collection()
- DFS PartitionScope object in a StructuredQuery implementation

### Use DQL

You can route a DQL query to a specific collection in the following ways. By default, DFC does not generate DQL, but you can turn off XQuery generation. See “[Turning off XQuery generation to support DQL](#)” on page 318.

- Route an individual query using the DQL *in collection* clause to specify the target of a SELECT statement. Use one of the two following syntaxes.
  - Collection names are separated by underscores .

```
select attr from type SDC where ... enable(  
fds_query_collection_collection1_collection2_...)  
  
select attr from type SDC in collection ('collection1','collection2',...)
```

- Collection names are in quotation marks, separated by commas.

```
select r_object_id from dm_document search document contains 'report'  
in collection ('default') enable(return_top 10)
```

- Route all queries that meet specific criteria using a DQL hint in dfcdqlhints.xml `enable(fds_query_collection_collectionname)` where collectionname is the collection name. For more information on the hints file, refer to *OpenText Documentum Search Development Guide*. The following hints route queries for a specific type to a known target collection appended to FDS\_QUERY\_COLLECTION\_.

```
<RuleSet>  
  <Rule>  
    <Condition>  
      <From condition="any">  
        <Type>my_type</Type>  
      </From>  
    </Condition>  
    <DQLHint>ENABLE(FDS_QUERY_COLLECTION_MYTYPECOLLECTION)</DQLHint>  
  </Rule>  
</RuleSet>
```

## Implement a DFC query builder API

If you have created a BOF module that routes documents to specific collections, customize query routing to the appropriate collection. Call `addPartitionScope(source, collection_name)` in `IDfQueryBuilder`. See “[Building a query with the DFC search service](#)” on page 319.

## Turning off XQuery generation to support DQL

Add the following setting to `dfc.properties` on the DFC client application:

```
dfc.search.xquery.generation.enable=false
```

## 10.8 Debugging queries

You can debug queries by clicking a collection in xPlore administrator. Choose Execute XQuery for the target collection or the top-level collection for the repository.



**Note:** Do not use xadmin to rebuild an index or change files that xPlore uses. If you remove segments, your backups cannot be restored. This tool is not aware of xPlore configuration settings in `indexserverconfig.xml`.

## 10.9 Building a query with the DFC search service

In a Documentum environment, the DFC APIs encapsulate complex functionality like stemming, wildcards, fuzzy search, hit count, and facets. With the DFC search service, you can create queries for one or more full-text indexed or non-indexed Servers. With Federated Search Services (FS2) product, you can query external sources and the client desktop as well. The DFC interface IDfQueryBuilder provides a programmatic interface. You can change the query structure, support external sources, support asynchronous operations, change display attributes, and perform concurrent query execution in a federation.

IDfQueryBuilder in the package com.documentum.fc.client.search allows you to build queries that provide the following information:

- Source query (getRootExpressionSet)
- Source list (required, several scope methods)
- Max result count (setMaxHitCount)
- Container of source names
- Transient search metadata manager bound to the query
- Transient query validation flag
- Facet definition (addFacetDefinition)
- Specific folder location (addLocationScope)
- Target for a particular collection (addPartitionScope)
- Specify parallel execution across several collections
- Sorting (addOrderByAttribute and addASCIIOrderByAttribute)

For examples of query building, refer to *OpenText Documentum Search Development Guide*. For information on creating and configuring facets and processing facet returns, see the chapter "Facets."

## 10.10 Building a query with the DFS search service

In a Documentum environment, the DFC and DFS APIs encapsulate complex functionality like stemming, wildcards, fuzzy search, hit count, and facets.

DFS search services provide search capabilities against Documentum repositories, as well as against external sources Documentum Federated Search Services (FS2) server. For complete information on the DFS search service and code examples, refer to *OpenText Documentum Enterprise Content Services Reference*.

Searches can either be blocking or non-blocking, depending on the Search Profile setting. By default, searches are blocking. Non-blocking searches display results dynamically. Multiple successive calls can be made to get new results and the query status. The query status contains the status for each source repository, indicating if it is successful, if more results are expected, or if it failed with errors.

The cache contains the search results populated in background for every search. The cache clean-up mechanism is both time-based and size-based. You can modify the cache clean-up properties by editing the `dfs-runtime.properties` file.

A structured query defines a query using an object-oriented model. The query is constrained by a set of criteria contained in an `ExpressionSet` object. An ordered list of `RepositoryScope` objects defines the scope of the query (the sources against which it is run). `PartitionScope` objects target the query to specific collections.

#### ExpressionScope

##### Example 10-13: Building a structured query

The following example sets the repository name, folder path in the repository, whether to include subfolders, targets a specific collection, creates the full-text expression, and specifies object type.

```
private Query getScopedQuery ()  
{  
    StructuredQuery structuredQuery = new StructuredQuery();  
    RepositoryScope scope = new RepositoryScope();  
    scope.setRepositoryName(m_docbase);  
    scope.setLocationPath("/SYSTEM");  
    scope.setDescend(true);  
    scope.setExcluded(true);  
    structuredQuery.setScopes(Arrays.asList(scope));  
  
    PartitionScope pScope = new PartitionScope();  
    pScope.setRepositoryName(m_docbase);  
    pScope.setPartitionName("partitionName");  
    structuredQuery.setPartitionScopes(Arrays.asList(pScope));  
  
    ExpressionScope eScope = new ExpressionScope();  
    eScope.setRepositoryName(m_docbase);  
    final ExpressionSet expressionSet = new ExpressionSet();  
    expressionSet.addExpression(new FullTextExpression("OpenText"));  
    eScope.setExpressionSet(expressionSet);  
    structuredQuery.setExpressionScopes(Arrays.asList(eScope));  
  
    structuredQuery.addRepository(m_docbase);  
    structuredQuery.setObjectType("dm_document");  
    // Set expression  
    ExpressionSet expressionSet2 = new ExpressionSet();  
    expressionSet2.addExpression(new PropertyExpression(  
        "object_name", Condition.CONTAINS, new SimpleValue("test")));  
    structuredQuery.setRootExpressionSet(expressionSet2);  
    return structuredQuery;  
}
```



## 10.11 Building a DFC XQuery

You can build XQuery expressions using DFC. However, if you are familiar with search customizations using DFC, it may be easier to use the DFC search service to build queries (“[Building a query with the DFC search service](#)” on page 319).

Perform the following steps to use the DFC APIs for building an XQuery:

1. Get a DFC client object and an IDfXQuery object. See “[Get IDfXQuery object](#)” on page 321.
2. Set the query target to xPlore. See “[Set the query target](#)” on page 321.
3. Create an XQuery statement. See “[Create the XQuery statement](#)” on page 321.
4. Set query options. See “[Set query options](#)” on page 322.
5. Execute the query. See “[Execute the query](#)” on page 323
6. Retrieve results. See “[Retrieve the results](#)” on page 323

### Get IDfXQuery object

Create a new DFC client and get an IDfXQuery object:

```
IDfClientX clientx = new DfClientX();
IDfXQuery xquery = clientx.getXQuery();
```

### Set the query target

An XQuery expression can be run against the Documentum XML Store or against the xPlore server. There are two implementations of IDfXQueryTargets in the package com.documentum.xml.xquery: DfFullTextXQueryTargets for xPlore and DfStoreXQueryTargets for XML Store. The following example sets the xPlore target:

```
IDfClientX clientx = new DfClientX();
IDfXQueryTargets fttarget = clientx.getXQueryTargets(IDfXQueryTargets.DF_FULLTEXT)
```

### Create the XQuery statement

For Documentum search clients, the IDfXQuery interface in the package com.documentum.xml.xquery runs user-defined XQuery expressions against xPlore indexes. XQuery expressions submitted through the IDfXQuery interface use the xPlore native full-text security evaluation.

Create an XQuery expression to submit. The following example creates a query for a string in the contents:

```
IDfXQuery xquery = clientx.getXQuery();
xquery.setXQueryString("unordered(for $i in collection('/docbase1/DSS/Data')
where ( ( $i/dmftdoc/dmftmetadata/*[r_creation_date[. >= '2008-12-20T08:00:00']] ) and ...");
```

## Set query options

You can set query options in the query before calling execution. For more information on these options, refer to the javadocs for IDfXQuery in the package com.documentum.xml.xquery.

The following example sets timeout, batch size, turns on caching, and saves the execution plan for debugging.

```
IDfXQuery xquery = clientx.getXQuery();
xquery.setTimeout(10000);
xquery.setBatchSize(200);
xquery.setSaveExecutionPlan(true);
xquery.setCaching(true);
```

Options:

- Debugging:
  - Get and set client application name for logging
  - Get and set save execution plan to see how the query was executed
- Query execution:
  - Get and set result batch size. For a single batch, set to 0.
  - Get and set target collection for query
  - Get and set query text locale
  - Get and set parallel execution of queries
  - Get and set timeout in ms
- Security:
  - Get and set security filter fully qualified class name
  - Get and set security options used by the security filter
  - Get and set native security (false sets security evaluation in the Documentum Server)
- Results:
  - Get and set results streaming
  - Get and set results returned as XML nodes
  - Get and set spooling to a file
  - Get and set synchronization (wait for results)
  - Get and set caching
- Summaries:
  - Get and set return summary
  - Get and set return of text for summary

- Get and set summary calculation
- Get dynamic summary maximum threshold
- Gets and sets length of summary fragments
- Get summary security mode

## Execute the query

After you have set the target, options, and XQuery in the instance of IDfXQuery, you execute, passing in the DFC session identifier and the xPlore target.

```
xquery.execute(session, fttarget);
```

## Retrieve the results

You can get the results as an input stream from the instance of IDfXQuery. You pass in the DFC session identifier of the session in which you executed the query.

```
InputStream results = xquery.getInputStream(session);
```

## Get the query execution plan

The query plan can be useful to OpenText Global Technical Services for evaluating slow queries. The query plan shows which indexes were probed and the order in which they were probed. Use one of the following options to save or fetch the query plan:

- DFC query builder API.. save:  
IDfXQuery.setBooleanOption(IDfXQuery.FtQueryOptions.SAVE\_EXECUTION\_PLAN,true). retrieve: IDfXQuery.getExecutionPlan(session)
- iAPI. save:  
apply,c,NULL,MODIFY\_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,ftengine.  
retrieve: The query execution plan written to dsearch.log, which is located in the logs subdirectory of the WildFly deployment directory.
- xPlore search API. save: IDfXQuery.setSaveExecutionPlan(true). retrieve:  
IFtSearchSession.fetchExecutionPlan(requestId)

# 10.12 Building a query using xPlore APIs

You can build a query using xPlore XQuery APIs. If you are familiar with DFC or DFS applications, it may be easier for you to create a query using their APIs. See “[Building a query with the DFC search service](#)” on page 319 or “[Building a query with the DFS search service](#)” on page 319.

Access to indexing and search APIs is through IDSearchClient in the package com.emc.documentum.core.fulltext.client. You execute a query by calling executeQuery for the interface com.emc.documentum.core.fulltext.client.IFtSearchSession. Each API is more fully described in the javadocs and in “[Search APIs](#)” on page 410.

Perform the following steps to use the xPlore APIs for query building:

1. Reference the jars in the SDK dist and lib directories in your classpath. Some of the classes in the jars will be used for later examples. For more information, see [“Setting up the xPlore SDK” on page 361](#).
2. Get a search session using IDSearchClient. The following example connects to the search service and creates a session.

```
public void connect() throws Exception
{
    String bootStrap = BOOT_STRAP;
    DSearchServerInfo connection = new DSearchServerInfo(m_host, m_port);
    IDSearchClient client = IDSearchClient.newInstance(
        "MySearchSession", connection);
    m_session = client.createFtSearchSession(m_domain);
}

private String m_host = "myhost";
private int m_port = 9300;
private String m_domain = "DSS_LH1";//this is the xPlore domain name
private IFtSearchSession m_session;
```

3. Create an XQuery statement. The following example creates a query for a string in the contents:

```
public void testQuery()
{
    String xquery = "for $doc in doc('/DSS_LH1/dsearch/Data/default') where
    $doc/dmftdoc[dmftcontents ftcontains 'strange'] return string(<R> <ID>{
    string($doc/dmftdoc/dmftmetadata//r_object_id)}</ID></R>)";
    executeQuery(xquery, options); //see "Executing a query"
}
```

4. Set query options. When you use an xPlore API to set options, the settings override the global configuration settings in the xPlore administration APIs. See the javadocs for IFtQueryOptions in the package com.emc.documentum.core.fulltext.common.search and [“Set the query target” on page 321](#). Add options like the following, and then provide the options object to the executeQuery method of IFtSearchSession. For example:

```
IFtQueryOptions options = new FtQueryOptions();
options.setSpooled(true);
```

5. Set query debug options. The enumeration FtQueryDebugOptions can be used to set debug options for IDfXQuery in DFC version 6.7 or higher. To set options, use the following syntax:

```
public String getDebugInfo(IDfSession session, FtQueryDebugOptions
debugOption)
throws DfException;
```

For example:

```
String queryid = xquery.getDebugInfo(m_session,
IDfXQuery.FtQueryDebugOptions.QUERY_ID);
```

6. Execute the query. See [“Execute the query” on page 323](#). Provide the query options and XQuery statement to your instance of IFtSearchSession.executeQuery, like the following:

```
requestId = m_session.executeQuery(xquery, options);
```

7. Retrieve results. The method executeQuery returns an instance of IFtQueryRequest from which you can retrieve results. See “[Retrieve the results](#)” on page 323.

The following example sets the query options, executes the query by implementing the IFtSearchSession method executeQuery, and iterates through the results, printing them to the console.

```
private void executeQuery (String xquery)
{
    String requestId = null;
    try
    {
        IFtQueryOptions options = new FtQueryOptions();
        options.setSpooled(true);
        options.setWaitForResults(true);
        options.setResultBatchSize(5);
        options.setAreResultsStreamed(false);
        requestId = m_session.executeQuery(xquery, options);

        Iterator<IFtQueryResultValue> results = m_session.getResultsIterator(
            requestId);
        while (results.hasNext())
        {
            IFtQueryResultValue r = results.next();
            System.out.print("results = ");
            //printQueryResult(r); See next step
            System.out.println();
        }
    }
    catch (FtSearchException e)
    {
        System.out.println("Failed to execute query");
    }
}
```

8. Retrieve results. Results from IFtSearchSession.executeQuery are returned as an instance of IFtQueryRequest from which you can retrieve results. Get each result value as an instance of IFtQueryResultValue by iterating over the IFtQueryRequest instance.

```
requestId = m_session.executeQuery(xquery, options);
Iterator<IFtQueryResultValue> results = m_session.getResultsIterator(
    requestId);
while (results.hasNext())
{
    IFtQueryResultValue r = results.next();
    printQueryResult(r);
}

private void printQueryResult(IFtQueryResultValue v)
throws FtSearchException
{
    if (v.getSelectListType().getType() !=
        IFtQuerySelectListItem.Type.NODE)
    {
        System.out.print(v.getValueAsString());
    }
    else
    {
        List<IFtQueryResultValue> children = (
            List<IFtQueryResultValue>) v.getValue();
        for (IFtQueryResultValue child : children)
        {
            printQueryResult(child);
        }
    }
}
```

## 10.13 Adding context to a query

Starting with DFC 6.7 SP1, you can add context information to a query. A Documentum client sets query context using the DFC search service or IDfXQuery.

Context information is not used to execute the query. The context information is available in audit events and reports. For example, query subscription and query warmup add context to indicate the type of query. For information on creating reports from the audit record, see “[Editing a report](#)” on page 351.

### DFC

IDfQueryProcessor method setApplicationContext(DfApplicationContext context). DfApplicationContext can set the following context:

- setApplicationName(String name)
- setQueryType(String type)
- setApplicationAttributes(Map<String, String> attributesMap). Set user-defined attributes in a Map object.

#### Example 10-14: DFC example

The following example sets the query subscription application context and application name. This information is used to report subscription queries.

Instantiate a query process from the search service, set the application name and query type, and add your custom attributes to the application context object:

```
    IDfQueryProcessor processor = m_searchService.newQueryProcessor(
        queryBuilder, true);

    DfApplicationContext anApplicationContext = new DfApplicationContext();
    anApplicationContext.setApplicationName("QBS");
    anApplicationContext.setQueryType("AUTO_QUERY");

    Map<String, String> aSetOfApplicationAttributes =
    new HashMap<String, String>();
    aSetOfApplicationAttributes.put("frequency", "300");
    aSetOfApplicationAttributes.put("range", "320");

    anApplicationContext.setApplicationAttributes(
    aSetOfApplicationAttributes);

    processor.setApplicationContext(anApplicationContext);
    IDfResultsSet results = processor.blockingSearch(60000);
```

The context is serialized to the audit record as follows:

```
<event name="AUTO_QUERY" component="search" timestamp=
2011-07-26T14:00:18-0700">
<QUERY_ID>PrimaryDsearch$706f93fa-e382-499c-b41a-239ae800da96
</QUERY_ID>
<QUERY>
<![CDATA[for $i in collection('/yttestenv/dsearch/Data')/dmftdoc[(
dmftmetadata//a_is_hidden = "false") and (dmftversions/iscurrent = "
```

```

        true") and (. ftcontains "xplore" with stemming using stop words
default)]
return string(<R>{$i//r_object_id}</R>)]]></QUERY>
<USER_NAME>unknown</USER_NAME>
<IS_SUPER_USER/>
<application_context>
<app_name>QBS</app_name>
<app_data>
<attr name="subscriptionid" value="080f444580029954" />
<attr name="frequency" value="300"/>
<attr name="range" value="320"/>
</app_data>
</application_context>
</event>

```

The event data is used to create a report. For example, a report that gets failed subscribed queries has the following XQuery expression. This expression gets queries for which the app\_name is QBS and the queries are not executed:

```

let $lib := '/SystemData/AuditDB/PrimaryDsearch/'
let $failingQueries := collection($lib)//event[name='AUTO_QUERY'
and application_context[app_name='QBS' and app_data[attr[
@name='frequency']/@value < attr[@name='range']/@value]]]/QUERY_ID
return $failingQueries

```

The result of this XQuery is the following:

```

<QUERY_ID>PrimaryDsearch$706f93fa-e382-499c-b41a-239ae800da96
</QUERY_ID>

```



## IDfXQuery

Use the API `FtQueryOptions` in the package  
`com.emc.documentum.core.fulltext.common.search`.

Call `setApplicationName(String applicationName)` to log the name of the search client application, for example, webtop.

Call `setQueryType(FtQueryType queryType)` with the `FtQueryType` enum.

## 10.14 Using parallel queries

The parallel query feature lets you run all queries against multiple collections in parallel, which improves query performance.

You can enable parallel query in one of the following ways:

- In `indexserverconfig.xml`, set the `query-parallel-execution` property to true:

```

<property name="query-parallel-execution" value="true" />

```

When this is set to true, a log message “Use Parallel Execution” will be recorded for each query logged in the `dsearch.log` file at the INFO level.

When there are too many queries running in parallel concurrently, a query may be forced to run in nonparallel mode to prevent system overload. In this case, the

following message is logged for the query at the INFO level: “Current active parallel thread count too high to allow parallel query execution. Running in nonparallel for the query”.

When the parallel search resource pool is exhausted, the system will attempt to retry the query request for up to the number of times specified by the query-executor-retry-limit property value in search-config. The amount of wait time between retry attempts is determined by the query-executor-retry-interval property setting.

- You can enable parallel queries in DFC by setting the following property to true in dfc.properties:

```
dfc.search.xquery.option.parallel_execution.enable = true
```

You can also use one of the following APIs to execute a query across several collections in parallel:

- DFC API: IDfXQuery.FTQueryOptions.PARALLEL\_EXECUTION
- xPlore API: IFtQueryOptions.setParallelExecution(true)



**Note:** Parallel queries may not perform better than a query that probes each collection in sequence.

## 10.15 Adding custom access to a thesaurus

Thesaurus expansion of queries is supported without customization. You can add custom access to a thesaurus that does not conform to the SKOS format. For example, you could add the Basitech Name Indexer to match people, places, or organizations.

To customize access, you must create a custom thesaurus class that implements IFtThesaurusHandler in the com.emc.documentum.core.fulltext.common.search package. Implement getTermsFromThesaurus(), which returns a collection of string terms from the thesaurus. Multi-term queries result in multiple calls to this method.

```
public Collection<String> getTermsFromThesaurus(Collection<String> terms,  
String relationship, int minValue, int maxValue);
```

Use the input terms from the query to probe the thesaurus.

You can use the optional XQuery *relationship* and *levels* parameters of *FTThesaurusOption* to specify special processing. For information on these parameters, see FTThesaurusOption in the XQuery and XPath Full Text 1.0 specification.

In the following example, the relationship value is “RT” (related term), and minValue and maxValue are 2:

```
'using thesaurus at 'thesaurusURI' relationship 'RT' exactly 2 levels'
```

In the following example, the relationship is “BT” (broader term), minValue is Integer.MIN\_VALUE, and maxValue is 2.

```
'using thesaurus at 'thesaurusURI' relationship 'BT' at most 2 levels'
```

## Setting up a custom thesaurus

Perform the following steps:

1. Implement a class that implements the IFtThesaurusHandler interface. See FASTThesaurusHandler.java for an example. This file is provided in the SDK at the following path: /samples/src/com/emc/documentum/core/fulltext/common/search/impl). A sample FAST thesaurus is provided at /samples/thesaurus.



**Note:** Only one instance of the class is created on startup, so multiple search threads share the class. Avoid thread synchronization issues or use thread local storage

2. Compile the custom class. Include dsearch-client.jar in your classpath when you compile. For example:

```
javac -cp dsearch-client.jar com\emc\documentum\core\fulltext\common\search\impl\SimpleThesaurusHandler.java
```

3. Package the class in a jar file and put it into the library *xplore\_home/*<wildfly\_version>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war/WEB-INF/lib. The path in the jar file must match the package name. For example:

```
jar cvf com\emc\documentum\core\fulltext\common\search\impl\dsearch-thesaurus.jar com\emc\documentum\core\fulltext\common\search\impl\SimpleThesaurusHandler.class
```

4. Modify indexserverconfig.xml to specify the custom thesaurus. Define a new thesaurus element under the domain that will use the custom thesaurus. Restart the xPlore instances after making this change. The following example indicates a thesaurus URI to a custom-defined class. When a query specifies this URI, the custom class is used to retrieve related terms.

```
<domain storage-location-name="default" default-document-category="" dftxml" name=... >
<collection ... >
...
<thesaurus uri="my_thesaurus" class-name="com.emc.documentum.core.fulltext.common.search.impl.FASTThesaurusHandler" />
</domain>
```

## Accessing the custom thesaurus in a query

You can specify custom thesaurus access in a DFC or IDfxQuery:

- DFC: setThesaurusLibrary(String uri). Use the URI that you defined in indexserverconfig.xml.
- DQL: Use the *ft\_use\_thesaurus\_library* hint.
- IDfxQuery: Add a thesaurus option. See FTThesaurusOption in the XQuery and XPath Full Text 1.0 specification. For example:

```
for $i score $s in collection('/testenv/dsearch/Data')
/dmftdoc[. ftcontains 'food products' using thesaurus default]
order by $s descending return $i/dmftinternal/r_object_id
```

You can access one thesaurus for full-text and one thesaurus for metadata. For example, you may have a metadata thesaurus that lists various forms of company names. The following example uses the default thesaurus to expand the full-text lookup and a metadata thesaurus to expand the metadata lookup:

```
IDfExpressionSet rootSet = queryBuilder.getRootExpressionSet();

//full-text expression uses default thesaurus
IDfFullTextExpression aFullTextExpression = rootSet.addFullTextExpression(
    fulltextValue);
aFullTextExpression.setThesaurusSearchEnabled(true);

//simple attribute expression uses custom metadata thesaurus
IDfSimpleAttributeExpression aMetadataExpression =
rootSet.addSimpleAttrExpression("companyname", IDfValue.DF_STRING,
IDfSimpleAttrExpression.SEARCH_OP_CONTAINS, false, false,
companyNameValue);
aMetadataExpression.setThesaurusSearchEnabled(true);
aMetadataExpression.setThesaurusLibrary(
    "http://search.opentext.com/metadatathesaurus");
```

## Using the thesaurus handler class

The FASTThesaurusHandler class is a sample implementation of the IFtThesaurusHandler interface (this class is included in the xPlore SDK.). When the class is instantiated by xPlore, it reads a FAST dictionary file and stores the term mappings into memory. This provides quick lookups to return related words from the FASTThesaurusHandler class when you run a query.

This file is provided in the SDK at the following path: /samples/src/com/emc/documentum/core/fulltext/common/search/impl). A sample FAST thesaurus is provided at /samples/thesaurus.

```
package com.emc.documentum.core.fulltext.common.search.impl;
import com.emc.documentum.core.fulltext.common.search.IFtThesaurusHandler;
import java.util.*;
import java.io.*;

public class FASTThesaurusHandler implements IFtThesaurusHandler
{
    public Collection<String> getTermsFromThesaurus(
        Collection<String> terms, String relationship,
        int minValue, int maxValue)
    {
        Iterator<String> termIterator = terms.iterator();
        Collection<String> result = new ArrayList<String>();
        while (termIterator.hasNext())
        {
            String key = termIterator.next();
            if (s_thesaurus.containsKey(key))
                result.addAll(s_thesaurus.get(key));
        }
        return result;
    }

    private static Map<String, Collection<String>> s_thesaurus =
        new HashMap<String, Collection<String>>();
    static
```

```

{
    try
    {
        String location = System.getenv("DOCUMENTUM") + "
/DocumentumThesaurusFAST.txt";
        FileInputStream fstream = new FileInputStream(location);
        DataInputStream in = new DataInputStream(fstream);
        BufferedReader br = new BufferedReader(new InputStreamReader(in));

        String line;
        while((line = br.readLine()) != null)
        {
            if (line.length() > 0)
            {
                String[] mapping = line.split("=", 2);
                String key = mapping[0];
                String related = mapping[1];
                // do some format checking
                if (key.length() < 1)
                    continue;
                // related should have at least "[?]", where ? is any character
                if (related.length() < 3)
                    continue;
                if (related.charAt(0) != '[' || related.charAt(related.length(
                )-1) != ']')
                    continue;
                related = related.substring(1, related.length()-1);
                String relatedTerms[] = related.split(",");
                Collection<String> terms = new ArrayList<String>(
                    relatedTerms.length);
                for (String term : relatedTerms)
                {
                    terms.add(term);
                }
                s_thesaurus.put(key, terms);
            }
        }
        catch (FileNotFoundException e)
        {
            System.out.println("FileNotFoundException while loading FAST Thesaurus: "
                + e.getMessage());
        }
        catch (IOException e)
        {
            System.out.println("IOException while loading FAST Thesaurus: "
                + e.getMessage());
        }
    }
}

```

## 10.16 Performing proximity searches

To search for a document that contains a word, for example “Texas”, within a number of words, for example “5”, of another term such as “parades”, you can enter proximity search parameters in the DA or Webtop Quick Search box. In this example, the search parameters are “(Texas AND parades) NEAR(5)”.

In DFC, you can search with full text expression:

```
IDfExpressionSet rootExpressionSet = queryBuilder.getRootExpressionSet();
rootExpressionSet.addFullTextExpression("(Texas AND parades) NEAR(5);")
```

The following xQuery constraint will be generated by DFC:

```
and (. ftcontains ("Texas" with stemming using stop words default ftand  
"parades" with stemming using stop words default distance at most 5 words))
```

# Chapter 11

## Facets

### 11.1 About Facets

Faceted search, also called guided navigation, enables users to explore large datasets to locate items of interest. You can define facets for the attributes that are used most commonly for search. Facets are presented in a visual interface, removing the need to write explicit queries and avoiding queries that do not return desired results. After facets are computed and the results of the initial query are presented in facets, the user can drill down to areas of interest. At drilldown, the query is reissued for the selected facets.

A facet represents one or more important characteristics of an object, represented by one or more object attributes in the Documentum object model. Multiple attributes can be used to compute a facet, for example, r\_modifier or keywords. Faceted navigation permits the user to explore data in a large dataset. It has several advantages over a keyword search or explicit query:

- The user can explore an unknown dataset by restricting values suggested by the search service.
- The data set is presented in a visual interface, so that the user can drill down rather than constructing a query in a complicated UI.
- Faceted navigation prevents dead-end queries by limiting the restriction values to results that are not empty.

Facets are computed on discrete values, for example, authors, categories, tags, and date or numeric ranges. Facets are not recommended for text fields such as content or object name, but are recommended to be used for fields having discrete values such as authors or categories. Facet results are not localized; the client application must provide localization.

Before you create facets, create indexes on the facet attributes. See “[Configuring facets in xPlore](#)” on page 334. Some facets are already configured by default.

For very specific use cases, if the out-of-the-box facet handlers do not meet your needs, you can define custom facet handlers for facet computation. For example, if a facet potentially include many distinct values, you can define ranges to group the values.

#### Facets and security

Facets restrict results based on the xPlore security filter, so that users see only those documents for which they have permission. If security is evaluated in the Documentum Server and not in xPlore, facets are disabled.

## API overview

Your search client application can define a facet using the DFC query builder API or DFS search service. For information on using the DFC query builder API, see “[Building a query with the DFC search service](#)” on page 319. For information on using the DFS search service, see “[Building a query with the DFS search service](#)” on page 319. Define custom facet handlers using the xPlore facet handler API, see “[Defining a facet handler](#)” on page 341. In most cases, the out-of-the-box facet handlers are sufficient.

Facets are computed in the following process. The APIs that perform these operations are described fully in the following topics. For facets javadocs, see the DFC or DFS javadocs.

1. DFC or DFS search service evaluates the constraints and returns an iterator over the results.
2. Search service reads through the results iterator until the number of results specified in `query-max-result-size` has been read (default: 10000).
3. For each result, the search service gets the attribute values and increment the corresponding facet values. Subpath indexes speed this lookup, because the values are found in the index, not in the xDB pages.
4. The search service performs the following on the list of all facet values:
  - a. Orders the facet values.
  - b. Keeps only the top facet values according to `setMax` (DFC) or `setMaxFacetValues` (DFS). Default: 10.
  - c. Returns the facets values and top results.

## 11.2 Configuring facets in xPlore

Facets are configured in `indexserverconfig.xml`. Your DFC-based application must also define the facet using query builder APIs. See “[Building a query with the DFC search service](#)” on page 319.

### Preconfigured facets in xPlore

The following facets are configured in `indexserverconfig.xml`. Do not add them to the application. (They are configured in a sub-path element whose `returning-contents` attribute is set to true.)

- `a_application_type` (Documentum application)
- `a_content_type` (file format)
- `a_gov_room_id` (room ID)
- `acl_domain` (permission set owner)
- `acl_name` (permission set)

- city (Content Intelligence Services attribute for CenterStage)
- company (Content Intelligence Services attribute)
- continent (Content Intelligence Services attribute for CenterStage)
- country (Content Intelligence Services attribute for CenterStage)
- keywords (user-defined list of keywords)
- location (CenterStage property)
- owner\_name (document owner)
- person (Content Intelligence Services attribute)
- r\_full\_content\_size (document size)
- r\_modifier (last person who modified the document)
- r\_modify\_date (last modification date)
- r\_object\_type (object type)
- state (Content Intelligence Services attribute for CenterStage)

## Configuring your own facets

For each attribute that is used as a facet, configure a subpath in indexserverconfig.xml.

1. Edit indexserverconfig.xml as described in “[Modifying indexserverconfig.xml on page 54](#).
2. Set the *returning-contents* attribute to true to compute facets on this attribute.
3. To configure the attribute datatype, set the *type* attribute to: `string`, `integer`, `boolean`, `datetime`, or `double`.

In the following example, `r_modifier` is available for use as a facet:

```
<path-value-index path=...>
  ...
  <sub-path returning-contents="true" compress="true" type="string"
    path="dmftmetadata//r_modifier"/>
</path-value-index>
```

For more information on subpath configuration, see “[Subpaths](#)” on page 176.

## 11.3 Creating a DFC facet definition

The class DfFacetDefinition in the DFC package com.documentum.fc.client.search represents a facet. You can set the following optional parameters using class methods. The valid values for these methods are dependent on the datatype of the facet (underlying attribute). See “[Facet datatypes](#)” on page 336.

- `setGroupBy(String)`: Group by strategy, depending on data type.
- `setMax(int)`: Maximum number of facet values for the facet. Default: 10. A value of -1 specifies unlimited number of facet values.
- `setName(String)`: Name of facet
- `setProperty(String)`: Used to set properties like timezone (for date type), range (for numeric type), and caseInsensitive (for string type).

The IDfQueryBuilder API `setMaxResultsForFacets()` limits the overall number of query results that are used to compute facets. The property `query-facet-max-result-size` in `indexserverconfig.xml` is equivalent to the API `setMaxResultsForFacets`. This setting controls the maximum number of results that are used to compute all facets in a single query.

The facet definition `setMax` API controls the size of the output in facets computation. For example, if a user has an average of 5 documents, a `setMaxResultsForFacets` value of 50 could return the top 10 users.

## 11.4 Facet datatypes

Each facet datatype requires a different grouping strategy. You can set the following parameters for each datatype in the specified `DfFacetDefinition` method (DFC) or `FacetDefinition` object (DFS). The main facet datatypes are supported: string, date, and numeric.

### String facet datatype

A string type facet accepts the following parameters:

- Set the maximum number of facet values. Default: 10. A value of -1 specifies unlimited results. For example, a facet for `r_modifier` with a maximum of two returns only two values. Results of documents modified by the first two modifiers are returned, but results for additional modifiers are not returned. DFC: `setMax(Integer max)`, DFS: `setMaxFacetValues(int maxFacetValues)`
- Set the sort order. DFC: `setOrderBy(ORDER orderby)` DFS: `FacetSort` object. Values of the `ORDER` enum (DFC) and `FacetSort` field: `FREQUENCY` (default) | `VALUE_ASCENDING` | `VALUE_DESCENDING` | `NONE`
- Set grouping strategy. You can specify the following optional properties using `DfFacetDefinition.setProperty`:
  - Case sensitivity: By default, string facets are case sensitive. For example, if a result set has a document with the value `OpenText` and another with `opentext`,

two facets are returned, *OpenText* and *opentext*. Set facet computation to ignore case by setting the optional property *caseInsensitive* to *true*.

- Count null values as a facet: Set *nullValueFacet* to *true*.
- *keepDuplicateValues*: Documents with repeating value attributes can have duplicate attribute values. By default, duplicate entries are removed.
- *alpharange*: Group by range. Set a property *range* that specifies ranges, for example: *a:m,n:r,s:z*. Specify range using ASCII characters. Uses unicode order, not language-dependent order. For example:

```
myFacetDefinition.setProperty("range", "a:m,n:r,s:z");
```

Facets are returned as *IDfFacetValue*. Following is an example of the XML representation of returned facet string values:

```
<facet name='r_modifier'>
  <eleme count='5' value='user2' />
  <element count='3' value='user1' />
</facet>
```

## Date facet datatype

A date type facet accepts the following parameters:

- Set the grouping strategy. When no value for max is specified, the most recent facets are returned first. DFC and DFS: *setGroupBy(String groupBy)*. Valid values are: day, week, month, quarter, year, *relativeDate* (Microsoft Outlook style). You can set the following grouping options:
  - *skipEmptyValues*: Call *setProperty(boolean skipEmptyValues)* for the facet definition. This optional property has a default value of *false*: The query returns all facet values between the oldest non-zero facet value and the most recent non-zero value (including zero values). If *true*, only the facet values with a count greater than zero are returned.
  - Set the time zone: Valid values for client time zone are expressed in UTC (relative to GMT), for example, *GMT+10*. To set the time zone in DFC, call *setProperty(String timezone)* for the facet definition. To set the time zone in DFS, call *setProperties(PropertySet set)* for a *FacetDefinition* with a *Property* having a UTC value.
  - Set the sort order. A *relativeDate* facet must set the order by parameter to *NONE*. Values of the *ORDER* enum: Values of the *ORDER* enum (DFC) and *FacetSort* field: *FREQUENCY* (default) | *VALUE\_ASCENDING* | *VALUE\_DESCENDING* | *NONE*
- DFC: *setOrderBy(ORDER orderby)*
- DFS: *FacetSort* object.
- Set the maximum number of facet values. Default: 10. A value of -1 specifies unlimited results. For example, a facet for *r\_modification\_date* with a maximum of two returns only two values. Results of documents modified for the first two modification dates are returned, but results for additional dates are not returned.
- DFC: *setMax(Integer max)*. DFS: *setMaxFacetValues(Integer maxFacetValues)*

Following is an example of the XML representation of returned facet date values:

```
<facet name='r_modify_date'>
  <elem count='5' value='2000-05-04T00:00:00'>
    <prop name="lowerbound">2000-05-04T00:00:00</prop>
    <prop name="upperbound">2000-05-05T00:00:00</prop>
  </elem>
  <elem count='3' value=' 2000-05-03T00:00:00'>
    <prop name="lowerbound">2000-05-03T00:00:00</prop>
    <prop name="upperbound">2000-05-04T00:00:00</prop>
  </elem>
</facet>
```

## Numeric facet datatype

A numeric type facet accepts the following parameters:

- Set the maximum number of facet values. Default: 10. A value of -1 specifies unlimited results. For example, a facet in DFC for r\_content\_size with setMax(2) has only two values. DFC: setMax(Integer max). DFS: setMaxFacetValues(int maxFacetValues)
- Set the sort order. A range must set the order by enum to NONE. Values of the ORDER enum: Values of the ORDER enum (DFC) and FacetSort field: FREQUENCY (default) | VALUE\_ASCENDING | VALUE\_DESCENDING | NONE. DFC: setOrderBy (ORDER orderby) DFS: FacetSort object.
- Group results in range order. To define the range, call setProperty(String range) for the facet definition (DFC) or call setProperties(PropertySet set) for a FacetDefinition with a Property (DFS). Valid values are a comma-separated list of ranges. Separate upper and lower bounds by a colon. The lower bound is inclusive, the upper bound exclusive. Ranges cannot overlap. A range can be unbounded, for example: '0:9,10:100,100:1000,1000:10000,10000:'. Default: none (treated as string).

DFC and DFS: setGroupBy(String groupBy). In the following example, a range property is defined and provided to setGroupBy. The definition in indexserverconfig.xml indicates that this type is integer.

```
DfFacetDefinition defAwardNumber = new DfFacetDefinition("dmftmetadata//award_no");
defAwardNumber.setOrderBy(DfFacetDefinition.ORDER.NONE);
defAwardNumber.setProperty("range", "1000:1009, 1010:1019, 1020:1029");
defAwardNumber.setGroupBy("range");
queryBuilder.addFacetDefinition(defAwardNumber);
```

Following is an example of the XML representation of returned facet numeric values for range=0:10,10:100,100:

```
<facet name='r_full_content_size'>
  <elem count='5' value='0:10'>
    <prop name='lowerbound'>0</prop>
    <prop name='upperbound'>10</prop>
  </elem>
  <elem count='3' value='10:100'>
    <prop name='lowerbound'>10</prop>
    <prop name='upperbound'>100</prop>
  </elem>
  <elem count='0' value='100:'>
    <prop name='lowerbound'>100</prop>
```

```
</elem>  
</facet>
```

## 11.5 Creating a DFS facet definition

You can use the DFS data model to create facets in a structured query. The following topics describe facet object and their place in a structured query.

### FacetValue

A FacetValue object groups results that have attribute values in common. The FacetValue has a label and count for number of results in the group. For example, a facet on the attribute `r_modifier` could have these values, with count in parentheses:

Tom Terrific (3)

Mighty Mouse (5)

A FacetValue object can also contain a list of subfacet values and a set of custom properties. For example, a facet on the date attribute `r_modify_date` has a value of a month (November). The facet has subfacet values of weeks in the specific month (Week from 11/01 to 11/08). xPlore computes the facet, subfacet, and custom property values.

### FacetDefinition

A FacetDefinition object contains the information used by xPlore to build facet values. The facet name is required. If no attributes are specified, the name is used as the attribute. Facet definitions must be specified when the query is first executed. A facet definition can hold a subfacet definition.

FacetSort is an enumeration that specifies the sort order for facet values. It is a field of the FacetDefinition object. The possible sort orders include the following: FREQUENCY (default) | VALUE\_ASCENDING | VALUE\_DESCENDING | NONE. A date facet must set the sort order to NONE.

### Adding facets to a structured query

Two fields in a StructuredQuery object relate to facets:

- List of facet definitions, returned by `getFacetDefinitions` and set by `setFacetDefinition`.
- Number of query results used by xPlore to compute the facets in a query, returned by `getMaxResultsForFacets` and set by `setMaxResultsForFacets`.

See *OpenText Documentum Enterprise Content Services Reference Guide* for more information on Query, StructuredQuery, QueryExecution, and SearchProfile.

### Facet results

A Facet object holds a list of facet values that xPlore builds.

A QueryFacet object contains a list of facets that have been computed for a query as well as the query ID and QueryStatus. This object is like a QueryResult object. A call to getFacets returns QueryResult.

The getFacets method of the SearchService object calculates facets on the entire set of query results for a specified Query. The method has the following signature:

```
public QueryFacet getFacets(  
    Query query, QueryExecution execution, OperationOptions options)  
throws SearchServiceException
```

This method executes synchronously by default. The OperationOptions object contains an optional SearchProfile object that specifies whether the call is blocking. For a query on several repositories that support facets, the client application can retrieve facets asynchronously by specifying a SearchProfile object as the OperationOptions parameter. Refer to *OpenText Documentum Enterprise Content Services* for more information on Query, StructuredQuery, QueryExecution, and SearchProfile.

You can call this method after a call to execute, using the same Query and queryId. Paging information in QueryExecution has no impact on the facets calculation.

## Paging facet results

Results can be retrieved from a specified index, page by page, as specified in the QueryExecution object: QueryExecution(startingIndex, maxResultCount, maxResultsPerSource). Paging of results is a new feature that FAST indexing did not support. Also, FAST returned only 350 results (configurable), whereas xPlore paging can support higher numbers of results.

The following results are obtained from a result set of 5000 with the specified paging parameters:

- (0, 25, 150): Gets the first page. Search retrieves 150 results but only the first 25 are returned. Results from 0 to 150 are cached. QueryStatus returns a hitCount of 5000.
- (25, 25, 150): Gets the next page. If the next page is no longer in the cache, search is launched to retrieve results from 25 to 50. Results from 25 to 50 are cached.
- (150, 25, 150): Gets a page. Search is launched to retrieve results from 150 to 175. Results from 150 to 300 are cached.
- (150, 151, 150): Gets a page. Results from 150 to 300 are returned. One result is missing because page size is greater than maxResultsPerSource. Results from 150 to 300 are cached.

### Example

```
// Get the SearchService  
ISearchService service = m_facade.getService(ISearchService.class, null,  
m_moduleName);  
  
// Create the query  
StructuredQuery query = new StructuredQuery();  
query.addRepository("your_docbase");
```

```

query.setObjectType("dm_sysobject");
ExpressionSet set = new ExpressionSet();
set.addExpression(new FullTextExpression("your_query_term"));
query.setRootExpressionSet(set);

// Add a facet definition to the query: a facet on r_modify_date
// attribute.
FacetDefinition facetDefinition = new FacetDefinition("date");
facetDefinition.addAttribute("r_modify_date");
// request all facets
facetDefinition.setMaxFacetValues(-1);
// group results by month
facetDefinition.setGroupBy("month");
// set sort order
facetDefinition.setFacetSort(FacetSort.VALUE_ASCENDING);
query.addFacetDefinition(facetDefinition);

// exec options: we don't want to retrieve results, we just want the
// facets.
QueryExecution queryExecution = new QueryExecution(0, 0);

// Call getFacets method.
QueryFacet queryFacet = service.getFacets(query, queryExecution,
    new OperationOptions());

// Can check query status: should be SUCCESS
QueryStatus status = queryFacet.getQueryStatus();
System.out.println(status.getRepositoryStatusInfos().get(0).getStatus());

// Display facet values
List<Facet> facets = queryFacet.getFacets();
for (Facet facet : facets)
{
    for (FacetValue facetValue : facet.getValues())
    {
        System.out.println(facetValue.getValue() + "/" +
                           facetValue.getCount());
    }
}

```

## 11.6 Defining a facet handler

Some facet handlers are available out-of-the-box in xPlore as part of the grouping strategies for facets, such as:

- Date ranges
- Numeric ranges
- Alphanumeric ranges

[“Facet datatypes” on page 336](#) describes the available grouping strategies for the main facet datatypes. If a custom facet belongs to one of these datatypes, you can use the out-of-the-box facet handlers. If the out-of-the-box facet handlers do not meet your requirements you can implement your own facet handler.

### Creating a custom facet handler

Implement the interface `IFacetFactory` in the package `com.emc.documentum.core.fulltext.indexserver.services.facets.handler` to define a custom facet handler.

1. Create a custom class that implements IFacetFactory.
2. To compile the classes, add dsearch-server.jar to your classpath. It located in <xplore\_home>/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war/WEB-INF/lib.
3. Package the custom handler classes as a jar
4. Copy the custom handler classes jar in xPlore war folder.
5. Edit indexserverconfig.xml as described in “[Modifying indexserverconfig.xml](#)” on page 54.

- a. Reference the class that implements IFacetFactory:

```
<search-config>
  ...
    <facet-handlers>
      <facet-handler class-name="my.package.MyFacetFactory1" />
      <facet-handler class-name="my.package.MyFacetFactory2" />
    </facet-handlers>
</search-config>
```

- b. If not already set, modify the subpath configuration for the facet as described in “[Configuring your own facets](#)” on page 335.

Reindexing is only required if you modify the subpath.

6. To use it in your application, reference the custom handler in the grouping strategy (GroupBy parameter) of the facet.

## 11.7 Sample DFC facet definition and retrieval

### Example 11-1: Sample facet definition

The DFC query builder interface IDfQueryBuilder creates facets with the addFacet(FacetDefinition) method. In the following example, a facet is created for the attribute r\_modifier.

First, set up session variables, and substitute values appropriate for your repository and instance owner:

```
private static final String DOCBASE = "DSS";
private static final String USER = "dmadmin";
private static final String PASSWORD = "dmadmin";
private static final long SEARCH_TIMEOUT = 180000;
```

Get a session and instantiate the search service and query builder:

```
IDfClient client = DfClient.getLocalClient();
IDfSessionManager m_sessionManager = client.newSessionManager();
DfLoginInfo identity = new DfLoginInfo(USER, PASSWORD);
m_sessionManager.setIdentity(DOCBASE, identity);

IDfSearchService m_searchService = client.newSearchService(
  m_sessionManager, DOCBASE);
IDfQueryManager queryManager = m_searchService.newQueryMgr();
IDfQueryBuilder queryBuilder = queryManager.newQueryBuilder("dm_sysobject");
```

Next, add the selected source and desired results:

```
queryBuilder.addSelectedSource(DOCBASE);
queryBuilder.addResultAttribute("r_object_id");
queryBuilder.addResultAttribute("object_name");
```

Start building the root expression set by adding the result attributes:

```
IDfExpressionSet exprSet = queryBuilder.getRootExpressionSet();

final String DATE_FORMAT = "yyyy-MM-dd'T'HH:mm:ss";
queryBuilder.setDateFormat(DATE_FORMAT);

exprSet.addSimpleAttrExpression("r_modify_date", IDfAttr.DM_TIME,
IDfSimpleAttrExpression.SEARCH_OP_GREATER_EQUAL, false, false, "
1980-01-01T00:00:00");

exprSet.addSimpleAttrExpression("r_modify_date", IDfAttr.DM_TIME,
IDfSimpleAttrExpression.SEARCH_OP_LESS_EQUAL, false, false, "
2010-01-01T00:00:00");
```

The previous code builds a query without facets. Now for the facets definition that defines a facet for person who last modified the document:

```
DfFacetDefinition definitionModifier = new DfFacetDefinition("r_modifier");
queryBuilder.addFacetDefinition(definitionModifier);
```

Another facet definition adds the last modification date and sets some type-specific options for the date:

```
DfFacetDefinition definitionDate = new DfFacetDefinition("r_modify_date");
definitionDate.setMax(-1);
definitionDate.setGroupBy("year");
queryBuilder.addFacetDefinition(definitionDate);
```

A subpath definition in indexserverconfig.xml defines a facet for keywords as follows:

```
<sub-path ...path="dmftmetadata//keywords" />
```

Keywords facet:

```
DfFacetDefinition definitionKeywords = new DfFacetDefinition("keywords");
queryBuilder.addFacetDefinition(definitionKeywords);
```

To submit the query and process the results, instantiate `IDfQueryProcessor`, which is described in the following topic.



#### Example 11-2: Getting facet values from `IDfQueryProcessor`

The `IDfQueryProcessor` method `getFacets()` provides facets results.

When several repositories return facets, the facets are merged. The merged results conform to the facet definition of maximum number of results and sort order. If you call `IDfQueryProcessor.getFacets()` before all sources finish query execution, the results can differ from final results.

First, instantiate the processor and launch the search:

```
IDfQueryProcessor processor = m_searchService.newQueryProcessor(  
queryBuilder, true);  
  
try  
{  
    processor.blockingSearch(SEARCH_TIMEOUT);  
} catch (Exception e)  
{  
    e.printStackTrace();  
}
```

For debugging, you can check the query status:

```
System.out.println("processor.getQueryStatus() = " + processor.getQueryStatus());  
System.out.println("processor.getQueryStatus().getHistory() = " +  
    processor.getQueryStatus().getHistory());
```

Get the non-facets results by calling getResults:

```
IDfResultsSet results = processor.getResults();  
for (int i = 0; i < results.size(); i++)  
{  
    IDfResultEntry result = results.getResultAt(i);  
    System.out.println(result.getId("r_object_id") + " = " + (result.hasAttr("object_name") ? result.getString("object_name") : "no title"));  
}
```

Get the facets results by calling getFacets:

```
List <IDfFacet> facets = processor.getFacets();  
for (IDfFacet facet : facets)  
{  
    System.out.println("--- Facet: " + facet.getDefinition().getName());  
    List<IDfFacetValue> values = facet.getValues();  
    for (IDfFacetValue value : values)  
    {  
        System.out.println("value = " + value);  
    }  
}
```



## 11.8 Tuning facets

### Limiting the number of facets to save index space and computation time

Every facet requires a special index, and every query that contains facets requires computation time for the facet. As the number of facets increases, the disk space required for the index increases. Disk space depends on how frequently the facet attributes are found in indexed documents. As the number of facets in an individual query increase, the computation time increases, depending on whether the indexes are spread out on disk.

### Limiting the number of results used to compute a facet

You can limit the number of results that are used to compute an individual facet. This setting varies the specificity of a facet. The setting depends on how many possible values a facet attribute can have. For example, for the Documentum

attribute `r_modifier`, you can have 10,000 users but wish to return only the top 10. If each user has an average of 5 documents, a `setMaxResultsForFacets` value of 50 could return the top 10 users. The computation will stop after 50 results are obtained. The 50 documents can belong to only five users, or they can belong to one user who contributes many documents. Set this property in the client application that issues the query.

## 11.9 Logging facets

To turn on logging for facets, use xPlore administrator and open the `dsearch-search` family. Set `com.emc.documentum.core.fulltext.indexserver.services.facets` to `DEBUG`:

Output is like the following:

```
2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- Begin facet computation

2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- Facets computed using 13 results.

2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- Facet handler string(r_modifier) returned 11 values.

2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- Facet handler string(r_modify_date) returned 4 values.

2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- Sort facets

2009-08-05 14:37:18,953 DEBUG [pool-3-thread-10]
c.e.d.c.fulltext.indexserver.services.facets.impl.CompositeFacetsProcessor
- End facet computation
```

## 11.10 Troubleshooting facets

*A query returns no facets*

Check the security mode of the repository. Use the following IAPI command:  
`get,c,l,ftsearch_security_mode ... 1`

```
API> retrieve,c,dm_ftengine_config ... 0800007580000916
...
0800007580000916
API> get,c,l,ftsearch_security_mode
...
0
```

If the command returns a 0, as in the example, set the security mode to evaluation in xPlore, not the Documentum Server. Use the following IAPI command:

```
retrieve,c,dm_ftengine_config
set,c,1,ftsearch_security_mode
1
```

```
save,c,1  
reinit,c
```

# Chapter 12

## Using reports

### 12.1 About reports

Reports provide indexing and query statistics, and they are also a troubleshooting tool. See the troubleshooting section for CPS, indexing, and search for uses of the reports.

Statistics on content processing and indexing are stored in the metrics database. Statistics on queries and final merges are stored in the audit database. Use xPlore administrator to query these statistics. Auditing supplies information to reports on administrative tasks or queries (enabled by default). For information on enabling and configuring auditing, see “[Auditing collection operations](#)” on page 209.

To run reports, choose **Diagnostic and Utilities** and then click **Reports**. To generate Documentum reports that compare a repository to the index, see “[Using ftintegrity](#)” on page 94.

### 12.2 Types of reports

The following types of reports are available in xPlore administrator.

**Table 12-1: List of reports**

Report title	Description
Audit records for search component	If search auditing is enabled in <b>System &gt; Global configuration</b> , you can view and create audit reports. Filter for <i>query type</i> : interactive, subscription, warmup, test search, report, metrics, ftintegrity, consistency checker, or all.
Audit records for admin component	If admin auditing is enabled in <b>System &gt; Global configuration</b> , you can view and create audit reports.
Audit records for final merge	Displays the following detailed information about final merges: domain, collection, instance, type, trigger time, start time, finish time, wait time, process time, total time, and status.

Report title	Description
Average time of an object in each indexing stage per hour/hour original/day/month	Reports the average bytes and time for the following processing stages: CPS Queue, CPS Executor, CPS processing, Index Queue, Index Executor, Index processing, Status Queue, Status Executor Queue, and Status Update. This value indicates how long a document sitting in the status update queue after the document has completed indexing. The timing is affected by the following two CPS configuration parameters: <i>status-requests-batch-size</i> and <i>status-thread-wait-time</i> .
Content too large to index summary report by domain	Displays format, count, average size, maximum size, and minimum size. Summarized by domain (Documentum repository).
Detailed query analysis	<ul style="list-style-type: none"> <li>• Performs a detailed query analysis and displays general statistics such as total, median, average, and maximum response times.</li> <li>• Performs analysis on queries that exceeded a specified amount of response time and displays detailed information (Query ID, User Name and Query Time).</li> <li>• Analyzes the impact of various factors such as FTDQL, Single-term, Multiterm, Wildcard queries on the query performance.</li> </ul>
Document processing error summary	Use first to determine the most common problems. Displays error code, count, and error text.
Document processing error detail	Drill down for error codes. Report for each code displays the request ID, domain, date and time, format, and error text.
Documents ingested per month/day/hour/minute	Totals for the corresponding period, including document count, bytes ingested, average processing latency, and CPS error count.
Get query text	Click the query ID from the report <i>Top N slowest queries</i> to get the XQuery expression.
Get object by ID	Gets object dftxml by object ID and domain.
QBS activity report by ID	Find the subscribed queries that take the longest processing time or are run the most frequently.
QBS activity report by user	Find users whose subscribed queries take the longest processing time.

Report title	Description
Query counts by user	For each user, displays domain, number of queries, average response time, and maximum and minimum response times, and last result count (sortable columns). Filter for <i>query type</i> : interactive, subscription, warmup, test search, report, metrics, ftintegrity, consistency checker, or all. <i>Number of users</i> sets last N users to display. To get slowest queries for a user, run the report <i>Top N slowest queries</i> .
Top query terms	Displays most common query terms including number of queries and average number of hits.
Top N slowest queries	Displays the slowest queries. Select <i>Number of results to display</i> . Optionally, specify a user to get slowest queries for a user. Specify the date and time range. Sort by various options, such as execution time, fetch time, total time, fetch count, and start time. Filter for <i>query type</i> : interactive, subscription, warmup, test search, report, metrics (query of metrics database for reports or indexing statistics), ftintegrity, consistency checker, or all.
User activity report	Displays query and ingestion activity and errors for the specified user and specified period of time. Data can be exported to Microsoft Excel. Query links display the xQuery.
IA message summary per hour/day/month	Displays the number of error messages and warning messages raised by document indexing failures during: <ul style="list-style-type: none"> <li>• each hour on a specified date</li> <li>• each day in a specified month</li> <li>• each month of a specified year</li> </ul>
IA message record	Displays detailed error and/or warning messages raised by document indexing failures during a specified period of time.

## 12.3 Document processing (CPS) reports

Run the *Document processing error summary report* to find the count for each type of problem. The error count for each type is listed in descending order. The following types of processing errors are reported: request and fetch timeout, invalid path, fetching errors, password protection or encryption, file damage, unsupported format, language and parts of speech detection, or document size.

View detailed reports for each type of processing error. For example, the Document processing error detail report for Error code 770 (File corrupt) displays object ID, domain, date, time, format, and error text. You can then locate the document in xPlore administrator by navigating to the domain and filtering the default collection for the object ID. Using the object ID, you can view the metadata in Documentum Server to determine the document owner or other relevant properties.

Run the report *Content Too Large to Index Summary Report by Domain* to see how many documents are being rejected for size. If your indexing throughput is acceptable, you can increase the size of documents being indexed. For more information about indexing performance, see “[Indexing performance](#)” on page 388.

Run the *User activity* report to see ingestion activity and error messages for ingestion by a specific user and time period.

## 12.4 Indexing reports

To view indexing rate, run the report *Documents ingested per month/day/hour*. The report shows *Average processing latency*. The monthly report covers the current 12 months. The daily report covers the current month. The hourly report covers the current day. From the hourly report, you can determine your period of highest usage. You can divide the document count into bytes processed to find out the average size of content ingested. For example, 2,822,469 bytes for 909 documents yields an average size of 3105 bytes. This size does not include non-indexable content.

## 12.5 Search reports

Enable auditing in xPlore administrator to view query reports (enabled by default).

### Top N slowest queries

Find the slowest queries by selecting *Top N slowest queries*. To determine how many queries are unselective, sort by *Fetch Count*. Results are limited by default in Webtop to 350.

Sort **Top N slowest queries** by *Hits Filtered Out* to see how many underprivileged users are experiencing slow queries due to security filtering. For information on changing the security cache, see “[Configuring the security cache](#)” on page 67.

## Get query text

To examine a slow or failed query by a user, get the query ID from *Top N slowest queries* and then enter the query ID into **Get query text**. Examine the query text for possible problems. The following example is a slow query response time. The user searched in Webtop for the string “xplore” (line breaks added here):

```
declare option xhive:fts-analyzer-class 'com.emc.documentum.core.fulltext.indexserver.core.index.xhive.IndexServerAnalyzer';
for $i score $s in collection(
/DSS_LH1/dsearch/Data') /dmftdoc[( ( (dmftmetadata//a_is_hidden = 'false') )
and ( (dmftinternal/i_all_types = '030a0d688000105') )
and ( (dmftversions/iscurrent = 'true') ) )
and ( (. ftcontains ( ('xplore') with stemming) ) ) ]
order by $s descending return
<dmrow>{if ($i/dmftinternal/r_object_id) then $i/dmftinternal/r_object_id
else
<r_object_id/>}{if ($i/dmftsecurity/ispublic) then $i/dmftsecurity/ispublic
else <ispublic/>}{if ($i/dmftinternal/r_object_type) then
$i/dmftinternal/r_object_type
else <r_object_type/>}{if ($i/dmftmetadata/*/owner_name)
then $i/dmftmetadata/*/owner_name
else <owner_name/>}{if ($i/dmftvstamp/i_vstamp) then $i/dmftvstamp/i_vstamp
else <i_vstamp/>}{if ($i/dmftsecurity/acl_name) then $i/dmftsecurity/acl_name
else <acl_name/>}{if ($i/dmftsecurity/acl_domain) then $i/dmftsecurity/acl_domain
else <acl_domain/>}<score dmfttype='dmdouble'>{$s}</score>{xhive:highlight(
$i/dmftcontents/dmftcontent/dmftcontentref)}</dmrow>
```

## Query counts by user

Use *Query counts by user* to determine which users are experiencing the slowest query response times or to see queries by a specific user. You can filter by date and domain.

## User activity report

Use the *User activity* report to display queries by the specified user for the specified time. Data can be exported to Microsoft Excel. Click a query link to see the xQuery.



**Note:** This report can take a very long time to run. If you enter a short date range or a user name, the report runs much faster.

## 12.6 Editing a report

You can edit any of the xPlore reports. Select a report in xPlore administrator and click **Save as**. Specify a unique file name and title for the report. Alternatively, you can write a new copy of the report and save it to the following location:

`xplore_home/<wildfly_version>/server/DctmServer_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes/reports.`

To see the new report in xPlore administrator, click somewhere else in xPlore administrator and then click *Reports*.

Reports are based on the W3C XForms standard. For a guide to the syntax in a typical report, see “[Report syntax](#)” on page 352.

## Accessing the audit record

The audit record is stored in the xDB database for the xPlore federation. To view the entire audit record, drill down to the AuditDB collection in **Data Management > SystemData**. Click AuditDB and then click auditRecords.xml.

## Adding a variable

Reports require certain variables. The XForms processor substitutes the input value for the variable in the query.

1. Declare it.
2. Reference it within the body of the query.
3. Define the UI control and bind it to the data.

These steps are highlighted in the syntax description, “[Report syntax](#)” on page 352.

## 12.7 Report syntax

xPlore reports conform to the W3C XForms specification. The original report XForms are located in the following directory:

*xplore\_home/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes/reports.*

You can edit a report in xPlore administrator and save it with a new name. Alternatively, you can copy the XForms file and edit it in an XML editor of your choice.

These are the key elements that you can change in a report:

**Table 12-2: Report elements**

Element	Description
xhtml:head/input	Contains an element for each input field
xhtml:head/query	Contains the XQuery that returns report results
xforms:action	Contains <i>xforms:setvalue</i> elements.
xforms:setvalue	Sets a default value for an input field. The <i>ref</i> attribute specifies a path within the current XForms document to the input field.
xforms:bind	Sets constraints for an input field. The <i>nodeset</i> attribute specifies a path within the current XForms document to the input field.
xhtml:body	Contains the XHTML markup that is rendered in a browser (the report UI)

The following example highlights the user the input field *startTime* in the report Query Counts By User (rpt\_QueryByUser.xml). The full report is line-numbered for reference in the example (some lines deleted for readability):

```
...<xforms:model><xforms:instance><ess_report xmlns="">
<input>
  <startTime/><endTime/>...
</input>
```

- 1 Defines an XForms instance, model (data definition), and ess\_report.
- 2 Model data: Declares the input fields for start and end date and time variables.

```
<query><![CDATA[ ...
let $u1 := distinct-values(collection('/SystemData/AuditDB'))/
event[@component = "search"...
and START_TIME[ . >= $startTime]...
return <report ...>...<rowset>...
for $d in distinct-values(collection('/SystemData...
and START_TIME[ . >= $startTime] and START_TIME[ . <= $endRange]]...
return let $k := collection('AuditDB')...and
START_TIME[ . >= $startTime] and START_TIME[ . <= $endRange]
...      return ...
} </rowset></report> ]]></query></ess_report></xforms:instance>
```

- 1 Specifies the XQuery for the report. The syntax conforms to the XQuery specification.

```
xhtml:head/xforms:model/xforms:instance/ess_report/query
```

- 2 let: Part of an XQuery FLWOR expression that defines variables. The first line specifies the collection for the report:

```
let $u1 := distinct-values(collection('/SystemData/AuditDB'))...
```

- 3 References the start time and end time variables and sets criteria for them in the query: as greater than or equal to the input start time and less than or equal to the input end time:

```
and START_TIME[ . >= $startTime]
and START_TIME[ . <= $endRange]]/USER_NAME)
```

- 4 return report/rowset: The return is an XQuery FLWOR expression that specifies what is returned from the query. The transform plain\_table.xsl, located in the same directory as the report, processes the returned XML elements.
- 5 This expression iterates over the rows returned by the query. This particular expression evaluates all results, although it could evaluate a subset of results.
- 6 This expression evaluates various computations such as average, maximum, and minimum query response times.
- 7 The response times are returned as row elements (evaluated by the XSL transform).

```
<xforms:action ev:event="xforms-ready">
  <xforms:setvalue ref="input/startTime" value="seconds-to-dateTime(
seconds-from-dateTime(local-dateTime()) - 24*3600)"/>...
</xforms:action>...
<xforms:bind nodeset="input/startTime" constraint="seconds-from-dateTime(
.) <= seconds-from-dateTime(..//endTime)"/>
```

```
<xforms:bind nodeset="input/startTime" type="xsd:dateTime"/>...
</xforms:model>...</xhtml:head>

<xhtml:body>...<xhtml:tr class="">
<xhtml:td>Start from:</xhtml:td>
<xhtml:td><xforms:group>
<xforms:input ref="input/startTime" width="100px" ev:event="DOMActivate">
<xforms:message ev:event="xforms-invalid" level="ephemeral">
The "Start from" date should be no later than the "to" date.
</xforms:message>
<xforms:action ev:event="xforms-invalid">
<xforms:setvalue ref="../endTime" ev:event="xforms-invalid"
value="../startTime"/><xforms:rebuild/>
</xforms:action>
</xforms:input></xforms:group></xhtml:td></xhtml:tr>...
```

1. *xforms:action* completes the xforms model data.
2. Sets the data mode using XPath expressions.
3. Binds the form control to the startTime variable and constrains the data that can be entered, using the XPath constraint *seconds-from-dateTime*.
4. Binds the form control to the XML schema datatype dateTime.
5. *xhtml:body*: Defines the UI presentation in xhtml. The body contains elements that conform to XForms syntax. The browser renders these elements.
6. The first table cell in this row contains the label **Start From**:
7. *xforms:input* contain elements that define the UI for this input control. Attributes on this element define the width and event that is fired.
8. *xforms:message* contains the message “when the entry does not conform to the constraint.
9. *xforms:action ev:event="xforms-invalid"* defines the invalid state for the input control. Entries after the end date are invalid.

## 12.8 Sample edited report

This example edits the *Query counts by user* report to add a column for number of failed queries.

1. Using an XML editor, open the report rpt\_QueryByUser.xml located in the following directory:  
*xplore\_home/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war/WEB-INF/classes/reports*.
2. Save the report with a new file name.
3. Change the report title in metadata/title, CDATA, to Failed Query Counts By User.
4. After the *column* element whose value is *Query Cnt*, add the following column:  
For example:<column type="integer">Failed Queries</column>

5. This step finds failed queries. Locate the variable definition for successful queries (*for \$j ...let \$k...*) and add your new query. Find the nodes in a QUERY element whose TOTAL\_HITS value is equal to zero to get the failed queries:

```
let $z := collection('AuditDB')//event[@component = "search" and @name = "QUERY" and START_TIME[ . >= $startTime and . <= $endRange] and USER_NAME = $j and TOTAL_HITS = 0]
```

6. Define a variable for the count of failed queries and add it after the variable for successful query count (*let \$queryCnt...*):

```
let $failedCnt := count($z)
```

7. Return the failed query count cell, after the query count cell (*<cell> { \$queryCnt } ...:*):

```
<cell> { $failedCnt } </cell>
```

8. Redefine the failed query variable to get a count for all users. Add this line after *<rowset...>let \$k...:*

```
let $z := collection('AuditDB')//event[@component = "search" and @name = "QUERY" and START_TIME[ . >= $startTime and . <= $endRange] and USER_NAME and TOTAL_HITS = 0]
```

9. Add the total count cell to this second rowset, after *<cell> { \$queryCnt } </cell>:*

```
<cell> { $failedCnt } </cell>
```

10. Save and run the report. The result is like the following:

The screenshot shows a report titled "My Query Counts By User". At the top, there is a toolbar with buttons for "View", "Run", "Open", and "Save As >>". Below the toolbar, the report title is displayed. A descriptive text explains the purpose of the report: "This report will summarize query statistics by User. This is meant to be helpful when trying to understand which users are seeing good or poor response. The admin can then further debug query response issue by looking at the top N slowest queries for a particular user." A "Search Options" section contains fields for "Start from" (set to 2010-06-01), "To" (set to 2010-06-16), "Display by Metric" (radio buttons for Time to First Result, Result Processing Time, and Total Response Time, with Time to First Result selected), and "Number of Results to Display" (set to 100). The main content area displays a table with the following data:

UserName	Query Cnt	Failed Queries	Avg Total Resp (msec)
admin	53	14	242.11
Administrator	1	0	1,375.00
unknown	1	0	328.00
Total over all 3 users	55	14	264.27

**Figure 12-1: Customized report for query count**

If your query has a syntax error, you get a stack trace that identifies the line number of the error. You can copy the text of your report into an XML editor that displays line numbers, for debugging.

If the query runs slowly, it will time out after about one minute. You can run the same query in the xDB admin tool.

## 12.9 Troubleshooting reports

If you update Internet Explorer or turn on enforced security, reports no longer contain content. Open **Tools > Internet Options** and choose the Security tab. Click Trusted sites and then click Sites. Add the xPlore administrator URL to the Trusted sites list. Set the security level for the Trusted sites zone by clicking **Custom level**. Reset the level to *Medium-Low*.

# Chapter 13

## Logging

### 13.1 Configuring logging

 **Note:** Logging can slow the system and consume disk space. In a production environment, run the system with minimal logging.

Basic logging can be configured for each service in xPlore administrator. Log levels can be set for indexing, search, CPS, xDB, and xPlore administrator. You can log individual packages within these services, for example, the merging activity of xDB. Log levels are saved to indexserverconfig.xml and are applied to all xPlore instances. xPlore uses logback and slf4j (Simple Logging Façade for Java) to perform logging.

To set logging for a service, choose **System Overview** in the left panel. Choose **Global Configuration** and then choose the **Logging** tab to configure logging for all instances. You can open one of the logging families like *xdb* and set levels on individual packages.

To customize the instance-level log setting, edit the logback.xml file in each xPlore instance. The logback.xml file is located in the WEB-INF/classes directory for each deployed instance war file, for example, *xplore\_home/<wildfly\_version>/server/DctmServer\_PrimaryDsearch/deployments/dsearch.war*. Levels set in logback.xml have precedence over log levels in xPlore administrator. Changes to logback.xml take up to two minutes to take effect.

Each logger logs a package in xPlore or your customer code. The logger has an appender that specifies the log file name and location. DSEARCH is the default appender. Other defined appenders in the primary instance logback configuration are XDB, CPS\_DAEMON, and CPS.

You can add a logger and appender for a specific package in xPlore or your custom code. The following example adds a logger and appender for the package *com.mycompany.customindexing*:

```
<logger name="com.mycompany.customindexing" additivity="false">
    <level>INFO</level>
    <appender name="CUSTOM" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>C:/xPlore/wildfly9.0.1/server/DctmServer_PrimaryDsearch/
            logs/custom.log
        </file>
        <encoder>
            <pattern>%date %-5level %logger{20} [%thread]
                %msg%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <rollingPolicy class="ch.qos.logback.core.rolling.
            FixedWindowRollingPolicy">
            <maxIndex>100</maxIndex>
            <fileNamePattern>C:/xPlore/wildfly9.0.1/server/DctmServer_
                PrimaryDsearch/logs/custom.log.%i</fileNamePattern>
        </rollingPolicy>
    </appender>
</logger>
```

```
</rollingPolicy>
<triggeringPolicy class="ch.qos.logback.core.rolling.
    SizeBasedTriggeringPolicy">
    <maxFileSize>10MB</maxFileSize>
</triggeringPolicy>
</appender>
</logger>
```

You can add your custom logger and appender to logback.xml. Add it to a logger family if you want your log entries to go to one of the logs in xPlore administrator. This is an optional step – if you don't add your custom logger to a logger family, it will still log to the file that you specify in your appender.

Logger families are defined in indexserverconfig.xml. They are used to group logs in xPlore administrator. You can set the log level for the family, or expand the family to set levels on individual loggers.

The following log levels are available. Levels are shown in increasing severity and decreasing amounts of information, so that TRACE displays more than DEBUG, which displays more than INFO.

- TRACE
- DEBUG
- INFO
- WARN
- ERROR

[“Troubleshooting the index agent” on page 111](#) provides information about the logging configuration for the index agent.

[“Enabling logging in a client application” on page 364](#) provides information about logging for xPlore client APIs.

## Viewing logs

You can view indexing, search, CPS, and xDB logs in xPlore administrator. Choose an instance in the tree and click **Logging**. Indexing and search messages are logged to dsearch. Click the tab for dsearch, cps, cps\_daemon, or xdb to view the last part of the log. Click **Download** to get links for each log file.

## Query logging

The xPlore search service logs queries. When you turn on query auditing (default is true), additional information is saved to the audit record and is available in reports. [“Auditing queries” on page 302](#) provides more information about query logging.

For each query, the search service logs the following information for all log levels:

- Start of query execution including the query statement
- Total results processed

- Total query time including query execution and result fetching

More query information is logged when native xPlore security (not Documentum Server security) is enabled. When query auditing is enabled, you can filter for the following query types in the search audit records report: interactive, subscription, warmup, test search, report, metrics, ftintegrity, consistency checker, or all.

Set the log level in xPlore administrator. Open **Services** in the tree, expand and select **Logging**, and click **Configuration**. You can set the log level independently for administration, indexing, search, and default. Levels in decreasing amount of verbosity: TRACE, DEBUG, INFO, WARN (default), and ERROR.

The log message has the following form:

```
2012-03-28 11:16:45,798 WARN [IndexWorkerThread-6]
c.e.d.c.f.i.core.index.plugin.XhivePlugin - Document id: 090023a380000202,
message: CPS Warning [Unknown error during text extraction(native code: 961,
native msg: access violation)].
```

To view a log, choose the instance and click **Logging**. The following examples from dsearch.log show a query:

```
2012-03-28 12:19:02,664 INFO [RMI TCP Connection(9)-10.8.47.144]
c.e.d.c.fulltext.indexserver.search.SearchServer -
QueryID=PrimaryDsearch$6f35b53d-34b8-470d-b699-5b4364ef0815,
query-locale=en,query-string=let $j:= for $i score $s in /dmftdoc
[. ftcontains 'ASMAgentServer' with stemming]
order by $s descending return <d> {$i/dmftmetadata//r_object_id}
{ $i/dmftmetadata//object_name } { $i/dmftmetadata//r_modifier } </d>
return subsequence($j,1,200) is running
...
2012-03-28 12:19:05,117 INFO [pool-14-thread-10]
c.e.d.c.f.i.admin.mbean.ESSAdminSearchManagement -
QueryID=PrimaryDsearch$6f35b53d-34b8-470d-b699-5b4364ef0815,
Result count=1,bytes count=187
```

## 13.2 CPS logging

CPS uses the xPlore logback and slf4j logging framework. A CPS instance that is embedded in an xPlore instance (installed with xPlore, not separately) uses the logback.xml file in WEB-INF/classes of the dsearch web application. A standalone CPS instance uses logback.xml in the CPS web application, in the WEB-INF/classes directory.

If you have installed more than one CPS instance on the same host, each instance has its own web application and logback.xml file. To avoid one instance log overwriting another, make sure that each file appender in logback.xml points to a unique file path.



## Chapter 14

# Setting up a Customization Environment

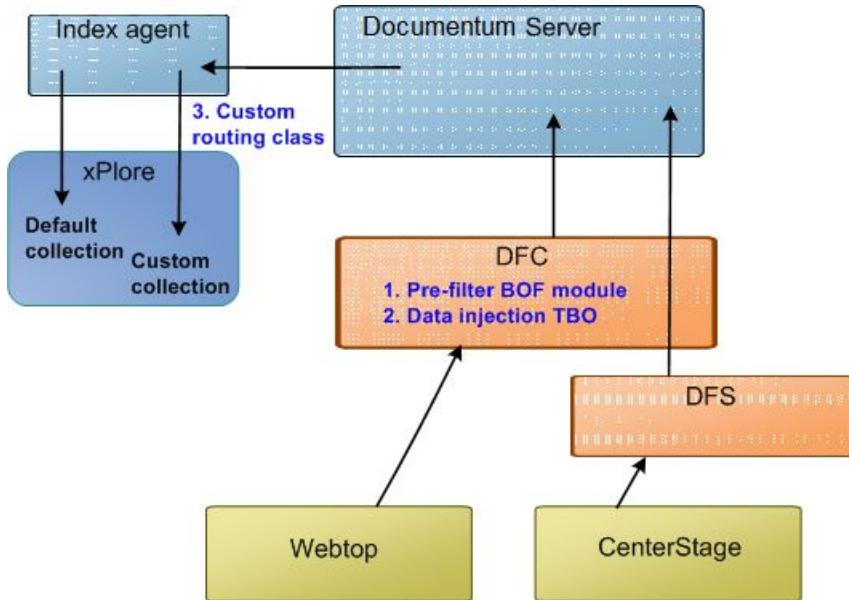
## 14.1 Setting up the xPlore SDK

1. Download the xPlore software development kit (SDK) from the OpenText My Support website.
2. Expand the download file to your development host. The SDK contains the following content:
  - README: Information about the SDK file.
  - LICENSE: The redistribution license for the Eclipse online help framework used by xPlore.
  - conf: Sample configuration files. The configuration parameters are described within the file dsearchclientfull.properties. For information on sample-logback-for-xplore-client.xml, see “[Enabling logging in a client application](#)” on page 364.
  - dist: xPlore distribution APIs and classes needed for customizations.
  - doc: This guide and javadocs.
  - lib: Java libraries that your application needs.
  - samples: Examples of a FAST thesaurus API and a natural language processing UIMA module.
3. Add the lib directory to your project or system classpath.
4. Add the distribution jar files to your build path.

## 14.2 Customization points

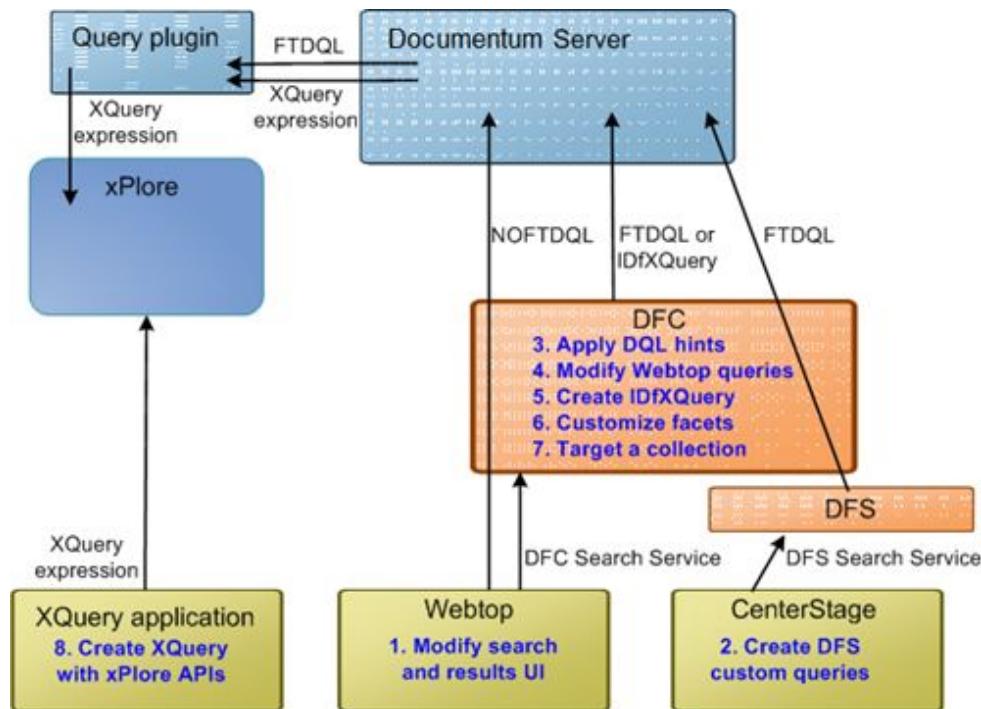
You can customize indexing and searching at several points in the xPlore stack. The following information refers to customizations that are supported in a Documentum environment.

The following diagram shows indexing customization points.

**Figure 14-1: Indexing customization points**

1. Using DFC, create a BOF module that pre-filters content before indexing.
2. Create a TBO that injects data from outside a Documentum repository, either metadata or content. You can use a similar TBO to join two or more Documentum objects that are related. See “[Injecting data and supporting joins](#)” on page 108.
3. Create a custom routing class that routes content to a specific collection based on your enterprise criteria. See “[Creating a custom routing class](#)” on page 187.

The following diagram shows query customization points.



**Figure 14-2: Query customization points**

1. Using WDK, modify Webtop search and results UI. See *OpenText Documentum Search Development Guide*.
2. Using DFS, implement StructuredQuery, which generates an XQuery expression. xPlore processes the expression directly. See “[Building a query with the DFS search service](#)” on page 319.
3. Using DFC or DFS, create NOFTDQL queries or apply DQL hints (not recommended except for special cases).  
DQL is evaluated in the Documentum Server. Implement the DFC interface IDfQuery and the DFS query service. FTDQL queries are passed to xPlore. Queries with the NOFTDQL hint or which do not conform to FTDQL criteria are not passed to xPlore. See “[DQL Processing](#)” on page 278.
4. Using DFC, modify Webtop queries. Implement the DFC search service, which generates XQuery expressions. xPlore processes the expression directly. See *OpenText Documentum Search Development Guide*.
5. Using DFC, create XQueries using IDfXQuery. See “[Building a DFC XQuery](#)” on page 321.
6. Create and customize facets to organize search results in xPlore. See the Facets chapter.
7. Target a specific collection in a query using DFC or DFS APIs. See “[Routing a query to a specific collection](#)” on page 317.

8. Use xPlore APIs to create an XQuery for an XQuery client. See “[Building a query using xPlore APIs](#)” on page 323.

## 14.3 Adding custom classes

Custom classes are registered in indexserverconfig.xml. To route documents from all domains using a custom class, edit this configuration file. Custom routing classes are supported in this version of xPlore.

1. Stop all instances in the xPlore system.
2. Edit indexserverconfig.xml in *xplore\_home/config* using an XML-compliant editor.
3. Add the following element to the root element index-server-configuration, between the elements *system-metrics-service* and *admin-config*, substituting your fully qualified class name.

```
<customization-config>
<collection-routing class-name="custom_routing_class_name" />
</customization-config>
```
4. Place your class in the indexagent.war WEB-INF/classes directory. Your subdirectory path under WEB-INF/classes must match the fully qualified routing class name.
5. Restart the xPlore instances, starting with the primary instance.

## 14.4 Enabling logging in a client application

Logging for xPlore operations is described in “[Configuring logging](#)” on page 357.

You can enable logging for xPlore client APIs and set the logging parameters in your client application. Use the file sample-logback-for-xplore-client.xml, which is located in the conf directory of the SDK. Save the file as logback.xml and put it in your client application classpath. All the xPlore JARs (JARs beginning with dsearch) will use the configuration.

The following procedure describes the high-level steps to use slf4j logger:

1. Import slf4j classes.
2. Use LoggerFactory.getLogger to get slf4j logger.
3. Log messages.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class <yourclass>
{
    private static Logger logger=
        LoggerFactory.getLogger(<yourclass>.class);
```

```
..  
logger.info("msg");
```

**Figure 14-3: Sample class using slf4j logger**

The slf4j documentation provides more information on slf4j logging.

## 14.5 Handling a NoClassDefFoundError exception

If you see the following Java exception, you have not included all of the libraries (jar files) in the SDK dist and lib directories:

```
...java.lang.NoClassDefFoundError: com/emc/documentum/fs/rt/ServiceException at  
com.emc.documentum.core.fulltext.client.admin.api.FtAdminFactory.getAdminService(  
FtAdminFactory.java:41)
```



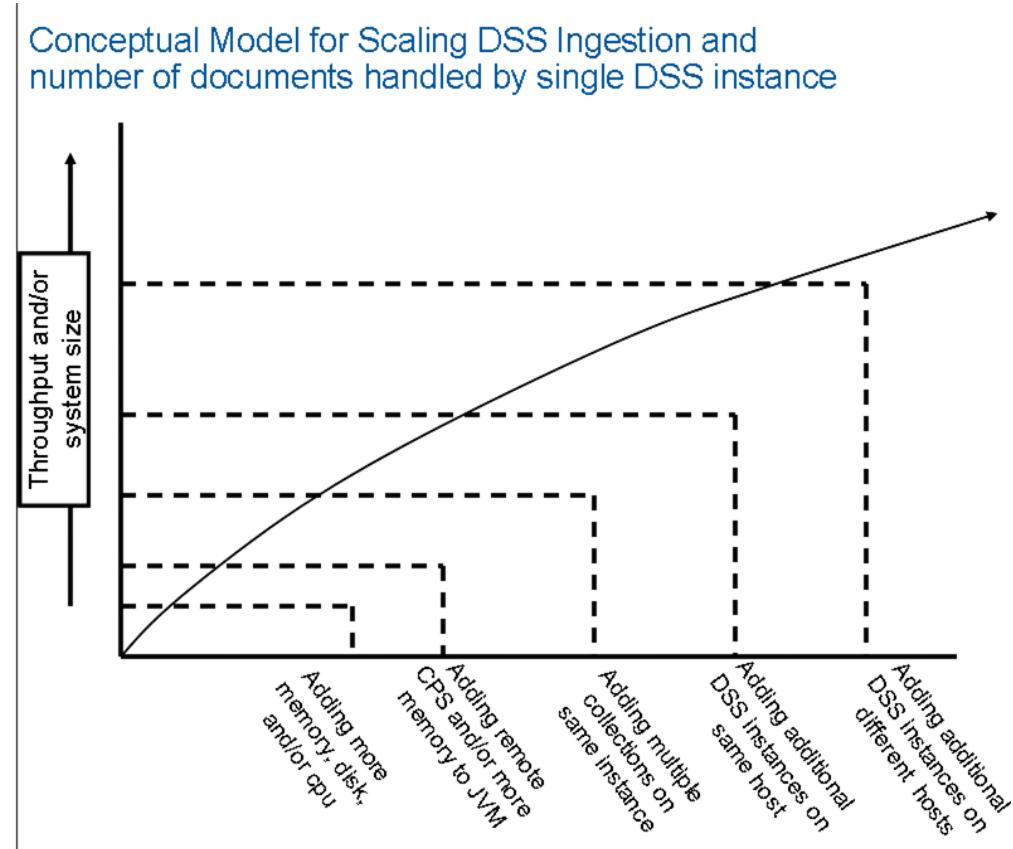
# Chapter 15

## Performance and Disk Space

### 15.1 Planning for performance

Plan your system sizing to match your performance and availability requirements. See the system planning topic in *OpenText Documentum xPlore Installation Guide*. This information helps you plan for the number of hosts and storage. The following diagram shows ingestion scaling. As you increase the number of documents in your system, or the rate at which documents are added, do the following:

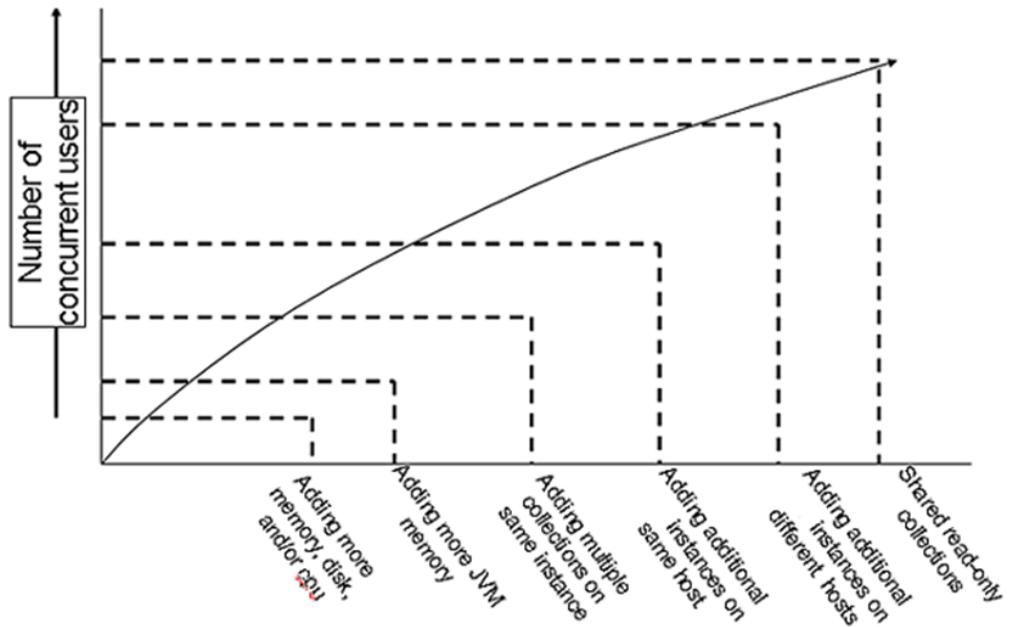
1. Add memory, disk, or CPU
2. Add remote CPS or more JVM memory
3. Increase the number of collections for ingestion specificity.
4. Add xPlore instances on the same or different hosts to handle your scaling needs.



**Figure 15-1: Scaling ingestion throughput**

Multiple collections increase the throughput for ingestion. You can create a collection, ingest documents to it, then move it to be a subcollection of a parent collection. See “[Moving a collection](#)” on page 202. Fewer collections speed up search.

Use the rough guidelines in the following diagram to help you plan scaling of search. The order of adding resources is the same as for ingestion scaling.



**Figure 15-2: Scaling number of users or query complexity in search**

## 15.2 Disk space and storage

### Managing xPlore disk space

As a rule of thumb, you need twice the final index space, in addition to the index space itself, for temporary Lucene merges. For example, if the index size after migration is 60 GB, you need an additional 120 GB for merges and optimizations. As the index grows, the disk space must grow correspondingly.

xPlore requires disk space for the following components. xDB and Lucene require most of the xPlore space. Choose a domain in xPlore administrator to get the total size.

**Table 15-1: How xPlore uses disk space**

Component	Space use	Indexing	Search
xDB	dftxml representation of document content and metadata, metrics, audit, and Document ACLs and groups.	Next free space consumed by disk blocks for batches of XML files.	Random access retrieval of particular elements and summary.

Component	Space use	Indexing	Search
Lucene	Stores an index of content and metadata. Locations: <i>storage-location</i> /data./lucene-index.	Information is updated through inserts and merges.	Inverted index lookup, facet and security lookup.
xDB transaction (redo) log	Stores transaction information.	Updates areas in xDB from log.	Sometimes provides snapshot during retrieval.
Lucene temporary working area	Used for Lucene updates of non-transactional data.	Uncommitted data is stored to the log. Allocate twice the final index size for merges. At least 20 GB of disk space should be allocated to this path for indexing to work properly.	None
Export file path	Configured in the CPS configuration file and used to store CPS processing results.	CPS processing results are stored in this area. At least 20 GB of disk space should be allocated to this path for indexing to work properly.	None

The following procedures limit the space consumed by xPlore:

- Saved tokens

If you have specified save-tokens for summary processing, edit indexserverconfig.xml to limit the size of tokens that are saved. For information on viewing and updating this file, see [“Modifying indexserverconfig.xml” on page 54](#). Set the maximum size of the element content in bytes as the value of the attribute *extract-text-size-less-than*. Tokens are not saved for larger documents. Set the maximum size of tokens for the document as the value of the attribute *token-size*. For details on extraction settings, see [“Configuring text extraction” on page 171](#).

- Lemmas

You can turn off alternative lemmatization support, or turn off lemmatization entirely, to save space. See [“Configuring indexing lemmatization” on page 133](#).

## Estimating index size (Documentum environments)

The average size of indexable content within a document varies from one document type to another and from one enterprise to another. You must calculate the average size for your environment. The easiest estimate is to use the disk space that was required for a Documentum indexing server with FAST. If you have not installed a Documentum indexing server, use the following procedure to estimate index size.

1. Perform a query to find the average size of documents, grouped by a\_content\_type, for example:

```
? ,c,select avg(r_full_content_size),a_content_type from dm_sysobject group by a_content_type order by 1 desc
```

2. Perform a query to return 1000 documents in each format. Specify the average size range, that is, r\_full\_content\_size greater than (average less some value) and less than (average plus some value). Make the plus/minus value a small percentage of the average size. For example:

```
? ,c,select r_object_id,r_full_content_size from dm_sysobject where r_full_content_size >(1792855 -1000) and r_full_content_size >(1792855 +1000) and a_content_type = 'zip' enable (return_top 1000)
```

3. Export these documents and index them into new, clean xPlore install.
4. Determine the size on disk of the dbfile and lucene-index directories in *xplore\_home./data*
5. Extrapolate to your production size.

For example, you have ten indexable formats with an average size of 270 KB from a repository containing 50000 documents. The Documentum Server footprint is approximately 12 GB. You get a sample of 1000 documents of each format in the range of 190 to 210 KB. After export and indexing, these 10000 documents have an indexed footprint of 286 MB. Your representative sample was 20% of the indexable content. Thus your calculated index footprint is 5 x sample\_footprint=1.43 GB (dbfile 873 MB, lucene-index 593 MB).

## Disk space vs. indexing rebuild performance

If you save indexing tokens for faster index rebuilding, they consume disk space. By default they are not saved. Edit `indexserverconfig.xml` and set `domain.collection.properties.property save-tokens` to true for a collection. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.

## Tuning xDB properties for disk space

You can set the following property in `xdb.properties`, which is located in the directory WEB-INF/classes of the specified xPlore instance. If this property is not listed, you can add it.

- `xdb.lucene.temp.path`

Temporary path for Lucene index. By default, it is pointed to the location where xPlore data is stored. If not specified, the current system property `java.io.tmpdir` is used.

## Adding storage

The data store locations for xDB libraries are configurable. The xDB data stores and indexes can reside on a separate data store, SAN or NAS. Configure the storage

location for a collection in xPlore administrator. You can also add new storage locations through xPlore administrator.

### Storage types and locations

**Table 15-2: Comparison of storage types performance**

Function	SAN	NAS	local disk	iSCSI	CFS
Used for Documentum Server	Common	Common (content)	Common	Rare	Rare
Network	Fiber	Ethernet	Local	Ethernet	Fiber
Performance	Best	Slower than SAN, improved with 10GE	Good until I/O limit reached	Slower than SAN, improved with 10GE	Almost as fast as SAN
High availability	Requires cluster technology	Provides shared drives for server takeover	Requires complete dual system	Requires cluster technology	Provides shared drives for server takeover
xPlore multi-instance	Requires network shared drives	Drives already shared	Requires network shared drives	Requires network shared drives	Drives already shared

## 15.3 System sizing for performance

You can size several components of an xPlore system for performance requirements:

- CPU capacity
- I/O capacity (the number of disks that can write data simultaneously)
- Memory for temporary indexing usage

### Sizing for migration from FAST

When you compare sizing of the FAST indexing system to xPlore, use the following guidelines:

- Size with the same allocations used for FAST, unless the FAST installation was very undersized or you expect usage to change.
- Use VMware-based deployments, which were not supported for FAST.
- Include sizing for changes to existing documents:
  - A modification to a document requires the same CPU for processing as a new document.
  - A versioned document requires the same (additional) space as the original version.

- Size for high availability and disaster recovery requirements.

### Add processing instances

CPS processing of documents is typically the bottleneck in ingestion. CPS also processes queries. You can add CPS instances either on the same host as the primary instance or on additional hosts (vertical and horizontal scaling, respectively). You should have at least one CPS instance for each Documentum repository. If you have process documents for multiple collections, add an instance for each collection.

A remote CPS instance does not perform as well as a CPS instance on an indexing instance. The remote instance adds overhead for the xPlore system. To add CPS instances, run the xPlore configuration script and choose **Create Content Processing Service Only**.

#### *Sizing for search performance*

You can size several components of an xPlore system for search performance requirements:

- CPU capacity
- Memory for query caches

Using xPlore administrator, change the value of *query-result-cache-size* in search service configuration and restart the search service.

## 15.4 Memory consumption

Following are ballpark estimates for memory consumption by the various components in an xPlore installation.

**Table 15-3: Average memory consumption**

Component	RAM
Index agent	1+ GB (4+ GB on 64-bit host)
xPlore indexing and search services	4 GB
CPS daemon	2 GB

For best performance, add index agent processes and CPS on hosts separate from the xPlore host.

## 15.5 Measuring performance

The following metrics are recorded in the metrics database. View statistics in xPlore administrator to help identify specific performance problems. Select an xPlore instance and then choose Indexing Service or Search Service to see the metric. Some metrics are available through reports, such as document processing errors, content too large, and ingestion rate.

**Table 15-4: Metrics mapped to performance problems**

Metric	Service	Problem
Ingestion throughput: Bytes and docs indexed per second, response time, latency	Indexing Service	Slow document indexing throughputs
Total number of documents indexed (or bytes)	Indexing Service	Indexing runs out of disk space
Formats	Content Processing Service	Some formats are not indexable
Languages	Content Processing Service	A language was not properly identified
Error count per collection	Content Processing Service	Finding collection where errors occurred
Number of new documents and updates	Content Processing Service	
Search response time	Search Service	Query timeouts or slow query response

To get a detailed message and count of errors, use the following XQuery in xPlore administrator:

```
for $i in collection('/SystemData/MetricsDB/PrimaryDsearch')
/metrics/record/Ingest[TypeOfRec='Ingest']/Errors/ErrorItem
return string(<R><Error>>{$i/Error}</Error><Space>" " </Space>
<Count>>{$i/ErrorCnt}</Count></R>
```

To get the total number of errors, use the following XQuery in xPlore administrator:

```
sum(for $i in collection('/SystemData/MetricsDB/PrimaryDsearch')
/metrics/record/Ingest [TypeOfRec='Ingest']/Errors/ErrorItem/ErrorCnt return $i)
```

## 15.6 Tuning the system

System tuning requires editing of indexserverconfig.xml. For information on viewing and updating this file, see “[Modifying indexserverconfig.xml](#)” on page 54.

### Excluding diacritics and alternative lemmas

Diacritics are not removed during indexing and queries. In some languages, a word changes meaning depending on a diacritic. You can turn off diacritics indexing to improve ingestion and query performance. See “[Handling special characters](#)” on page 138.

Alternate lemmas are also indexed. A word like *swim* is indexed as more than one part of speech (*swim* and *swimming*) is more likely to be found on search. You can turn off alternative lemmas to improve ingestion and query performance. See “[Configuring indexing lemmatization](#)” on page 133.

### Excluding xPlore files from virus scanners

Performance of both indexing and search can be degraded during virus scanning. Exclude xPlore directories, especially the *xplore\_home*/data directory.

### Tuning memory pools

xPlore uses four memory caches. The last three are part of the xPlore instance memory and have a fixed size:

- OS buffer cache
  - Holds temporary files and xDB data.
- xDB buffer cache
  - Stores XML file blocks for ingestion and query. Increase for higher query rates: Change the value of the property `xhive-cache-pages` in `indexserver-bootstrap.properties`. This file is located at `xplore_home/server/DocmServer_Node_name/deployments/dsearch.war/WEB-INF/classes`. Back up the xPlore federation after you change this file, and then restart all instances.
- Lucene working memory
  - Used to process queries. Lucene working memory is consumed from the host JVM process. Increasing the JVM memory usually does not affect performance.
- xPlore caches
  - Temporary cache to buffer results. Using xPlore administrator, change the value of `query-result-cache-size` in search service configuration and restart the search service.

### Tuning virtual environments

VMware deployments require more instances than physical deployments. For example, VMware is limited to eight cores.

## Sizing the disk I/O subsystem

xPlore supports local disk, SAN, and NAS storage. These storage options do not have equal performance. For example, NAS devices send more data and packets between the host and subsystem. Jumbo frame support is helpful as is higher bandwidth.

## Using compression

Indexes can be compressed to enhance performance. Compression uses more I/O memory. The *compress* element in indexserverconfig.xml specifies which elements in the ingested document have content compression to save storage space. Compressed content is about 30% of submitted XML content. Compression can slow the ingestion rate by 10-20% when I/O capacity is constrained. See “[Configuring text extraction](#)” on page 171.

Sample setting:

```
<compress>
  <for-element name="dmftcontentref"></for-element>
</compress>
```

## 15.7 Tuning Lucene internal merges

Multipath indexes are implemented using Apache Lucene (<http://lucene.apache.org>), a high-performance, open-source search library. xPlore stores multipath indexes separately in the xDB database as lucene blobs. Internally, a multipath index consists of Lucene segments, small chunks that contain one or more indexed documents. New Lucene segments are added as new documents are being indexed.

To improve query performance, Lucene automatically merges smaller segments into larger ones based on your settings. You can adjust the values of two xDB properties to fine tune the indexing performance.

- mergeFactor

*mergeFactor* decides the quantity of Lucene segments in one sub-index, which defaults to 10. Using a higher value of *mergeFactor* keeps more Lucene segments in one sub-index and further speeds up the indexing process. However, this may cause bad query performance.

- maxMergeDocs

Use the *maxMergeDocs* to specify the largest size of legitimate segments for merging. While merging segments, Lucene will ensure that no segment with more than *maxMergeDocs* is created. For example, if you set *maxMergeDocs* to 1000, when you add the 10,000th document, instead of merging multiple segments into a single segment of size 10,000, Lucene will create a 10th segment of size 1000, and keep adding segments of size 1000 for every 1000 documents added. A larger *maxMergeDocs* is better suited for batch indexing.



**Note:** When optimizing the quantity of segments, a final merge ignores the `maxMergeDocs` setting.

## 15.8 Managing sub-index merges

At the xPlore level, Lucene segments are logically organized into sub-indexes that represent updates to the index created by individual xDB transactions. xPlore indexes documents into xDB in batches, and each batch corresponds to an xDB transaction that creates a new sub-index. A query into a multipath index probes the union of all applicable sub-indexes. Over time, as increasingly more documents are indexed, the number of sub-indexes can become very large, which consumes more memory and disk space, and adversely impacts the indexing and query performance. To optimize system performance, smaller sub-indexes are periodically merged into larger ones, keeping the number of sub-indexes at a minimum, and ideally producing a single, optimized, large index (called the final index).

There are two distinct merging processes at play here: non-final merge and final merge. Both final and non-final merging tasks produce new sub-indexes as a result of merging smaller sub-indexes. The original sub-indexes will eventually be deleted by a periodic index cleaning task.

You can monitor the merge of a collection as described in “[Monitoring merges of index data](#)” on page 204.

### Non-final merge

Non-final merge is an asynchronous process that merges smaller sub-indexes into larger ones, which frees up memory and disk space and speeds up query. A non-final merge is more lightweight than a full final merge and has less impact on the indexing and query performance, so you can configure it to run more frequently. During a non-final merge, sub-indexes under a certain size (specified in the `nonFinalMaxMergeSize` property in `xdb.properties`) are merged into a fresh, new index at regular intervals (specified in the `cleanMergeInterval` property in `xdb.properties`).

### Final merge

Final merge is an asynchronous process that merges all eligible sub-indexes into a single, optimized “final” index. Although the desired outcome of a final merge is a single index, in practice, this is often not the case since during the final merge process, which takes quite a long time, new sub-indexes are created as a result of the ongoing indexing process.

A final merge is very CPU-intensive and I/O-intensive, and can take a long time to complete. At any point in time, only one final merge is allowed to run on a single node.

The final merge process also takes up a lot of disk space. During a final merge, the system shrinks existing sub-indexes by moving and consolidating index entries into

a new sub-index which resides in an empty luceneblob. When no empty luceneblob is not available, the system will yield this chance and wait for the next final merge to run. As a result, a final merge can require up to two times the size of the final index to move things around during the merging process. For example, you can see disk space usage at 100G at one point but 300G at another point when a final merge is in progress. Under such circumstances, you can manually launch a final merge to accelerate the merging process and free more disk space.

Final merge should not be run during performance-critical periods such as business hours, especially for large indexes.

The final merge policy is configurable but it requires care because final merge has a direct impact on the overall performance of the system. While it is desirable to keep the number of sub-indexes small, sub-index merging can be time-consuming, CPU-intensive, and I/O-intensive. If done too often, final merges may cause noticeable performance drop in both the indexing and query performance. Therefore, you should closely monitor and carefully schedule final merges and avoid them during performance-critical hours.

You can manage final merges in one of the following ways:

- Manually starting and stopping final merges

You can manually start a final merge on a collection directly from xPlore Administrator.

- Interval-based scheduling (instance-specific)

By default, a collection uses the Lucene internal final merge interval setting to run final merges at specified intervals. You can set the interval by specifying the `<xdb.lucene.finalMergingInterval>` value in `xdb.properties` and it applies to all the collections in the xPlore instance that uses this scheduling option.

Once set, the interval-based scheduler takes effect immediately. It runs the initial final merge and sets the current time as the starting point to schedule subsequent final merges based on the specified interval. However, the starting point is reset and final merges rescheduled whenever the instance the collection is bound to is restarted, which makes the final merge running schedules subject to change and not very predictable or manageable.

For example, at 20:00 on the first day, you set final merges to run on a collection every 24 hours. The first final merge runs immediately after the scheduling takes effect and the second final merge runs at 20:00 the next day. On the third day, the xPlore instance to which the collection is bound is restarted at 9:00. This resets the starting point of the scheduler for the collection to 9:00 and reschedules subsequent final merges to run at 9:00 every day. Do not use the interval-based scheduler if you want to maintain a precise final merge schedule.

- Cron-style scheduling

(collection-specific)

With this method, you can configure final merges to run at fixed times, dates, or intervals. Once set, the Cron-style scheduler settings takes effect immediately. Cron-style scheduling applies to the current collection only.

While Cron-style scheduling gives you more flexibility and precise control over when to start final merges, it requires you to have a good understanding of your system usage. Use this scheduling mode when you know exactly when is the best time to run final merges.

- Threshold-based scheduling

Through threshold-based scheduling, you can configure final merges to run periodically not only based on time, but also based on necessity and priority, taking into account the current state of sub-indexes. This advanced tuning tool allows you to balance the need to merge sub-indexes into the final index (too many sub-indexes impacts performance) and the need to avoid unnecessary or too-frequent final merges (final merges also impact performance) to achieve optimal system performance.

There is no user interface for configuring the threshold-based schedule. All the settings are configured through properties in `indexserverconfig.xml`.

The final merge scheduling methods are mutually exclusive. Only one scheduler is effective at a time and you cannot use the combination of multiple scheduler settings.

You can use the Audit Records for Final Merge report to view detailed final merge log data to quickly identify performance issues associated with final merges.

### 15.8.1 Manually starting and stopping final merges

The final merge is very I/O intensive and may cause noticeable performance drop in both the indexing and query performance, so you should avoid running final merges during performance-critical hours.

Before you start a final merge on a collection, we recommend that you stop other operations such as consistency check, document migration, backup and restore on the collection; otherwise, the final merge may significantly lower the performance of these operations.

To start a final merge manually:

1. Under Data Management in xPlore Administrator, go to the collection page and click the Start Merging button.
2. A confirmation dialog box appears warning you of the performance impact that the final merge operation may bring. Click OK to start final merge on the collection.

However, clicking Start Merging may not start the final merge immediately. If there is already another final merge running on the current xPlore instance, the final merge you manually started waits in a queue and will not run until the final merge in progress is completed or stopped, since at any point in time, only one final merge operation can run on an xPlore instance.

 **Example 15-1:**

Unlike scheduled final merges, manually started final merges are not affected by any blackout period settings.

If you find a final merge is in progress and is causing huge performance drops on your system—slowing ingestions and queries, or preventing backup—you can stop it immediately.

When a collection is merging, a Merging icon is displayed next to the collection in the Data Management view of the collection. To stop a final merge manually, click Stop Merging.

You can also start and stop a final merge using CLI commands. See “[Final merge CLIs](#)” on page 236.



## 15.8.2 Managing final merges through interval-based or Cron-style scheduling

Use the following steps to manage final merges using the interval-based or Cron-style scheduler:

1. In xPlore Administrator, under Data Management, click the collection.
2. On the collection page, click Configuration.
3. In the Edit Collection page, choose a Final Merge Scheduler option.
  - Instance-specific scheduler  
If you select this option, you can adjust the interval by specifying the `<xdb.lucene.finalMergingInterval>` value in `xdb.properties`.
  - Collection-specific scheduler  
Choose one of the following schedule formats and enter your values:
    - Fixed interval
    - Daily
    - Weekly
    - AdvancedIf you define a schedule in advanced format that equates to one of the simpler formats listed above, the simpler format will be selected instead after you save the settings. For example, if you enter 6 in the Day of week field and save the settings, you will see Weekly option selected with Sat checked when you review the settings.
4. Click Save. The scheduler is effective immediately.



**Note:** Final merge scheduling options are not available for collection with state search\_only or off\_line.

### 15.8.3 Final merging priorities and prioritization rules

There are three priority levels of final merges in threshold-based scheduling:

- High (must-merge)

Final merge must be run immediately on the collection to avoid index and query slowdowns.

- Medium (optional-merge)

Final merge is optional and can wait until a relatively less performance-critical time and until all high-priority final merges on the node are completed.

- Low (not-merge)

You do not need to run final merge on the collection because the performance impact of the final merge itself will outweigh the performance gain it brings.

The priority of a final merge is determined by two factors: the number of sub-indexes and the number of black nodes on the collection.

Too many sub-indexes, especially large ones, slow down indexing and query. The larger the number of sub-indexes, the more performance impact, and the higher the priority to run a final merge. When the quantity of sub-indexes reaches the must-merge threshold, it is important to run a final merge immediately to optimize the system performance. On the other hand, if there are too few sub-indexes and the number falls below a certain point (not-merge threshold), triggering a resource-intensive final merge is inefficient and will hurt performance.

You set the sub-index quantity thresholds through the following properties in `indexserverconfig.xml`:

- `lmpi-finalmerge-threshold-cleanentry-mustmerge`
- `lmpi-finalmerge-threshold-cleanentry-notmerge`

When an index entry is deleted, it is marked as deleted in the sub-index but is not physically removed until a final merge is run. Such index entries are called black nodes.

Too many black nodes take up disk space and slow down indexing and query. The larger the number of black nodes, the more impact on available disk space and performance, and the higher the priority to run a final merge. When the number of black nodes reaches the must-merge threshold, it is important to run a final merge immediately to optimize the system performance. However, if there are not many black nodes and the number falls below the not-merge threshold, triggering a resource-intensive final merge is inefficient and will lead to worse performance.

Set the black node quantity thresholds through the following properties within the *index-config* element in *indexserverconfig.xml*:

```
...
<index-config>
  <properties>
    <property value="50000" name="lmpi-finalmerge-threshold-bl-mustmerge">
    <property value="5000" name="lmpi-finalmerge-threshold-bl-notmerge">
    ...
  </properties>
...

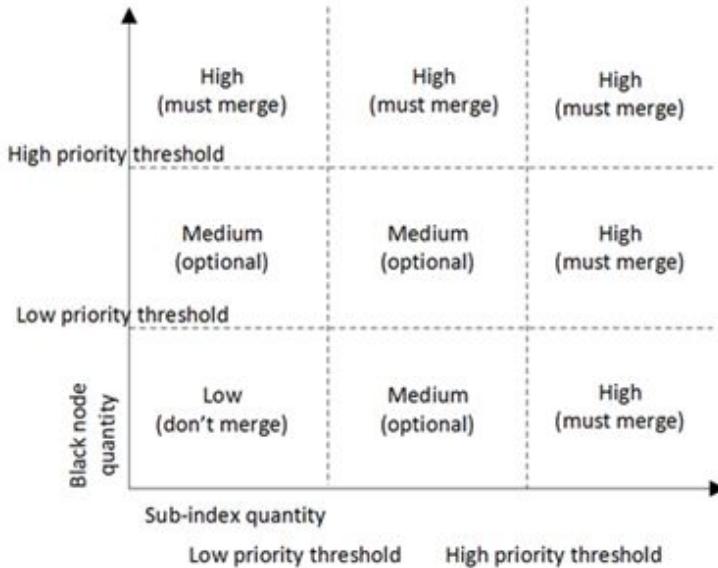
```

## Prioritizing a final merge

When prioritizing a final merge, xPlore retrieves real-time sub-indexes data on the collection and makes prioritization decisions based on all threshold settings.

High priority (must-merge) thresholds take precedence over low priority (not-merge) thresholds. As long as one high priority (must-merge) threshold is reached (either sub-index quantity or black node quantity), it is considered a high priority (must-merge) to run a final merge.

The final merge prioritization rules are illustrated as follows:



## 15.8.4 mergethruschedule

The threshold-based scheduler does not have a user interface. All the settings are configured in `indexserverconfig.xml`.

To configure the threshold-based scheduler:

1. Stop xPlore.
2. Enable the threshold-based scheduler by setting the following property within the `indexconfig` element to true in `indexserverconfig.xml`:

```
...
<index-config>
  <properties>
    <property value="true" name="lmpi-finalmerge-threshold-enabled">
      ...
    </properties>
...

```

The threshold-based scheduler is disabled by default. Once enabled, it overrides the interval-based scheduler or the Cron-style scheduler. When the threshold-based scheduler is disabled, you need to manually revert to the interval-based scheduler or the Cron-style scheduler through xPlore Administrator.

3. Configure the criteria for prioritizing final merges through properties within the `index-config` element in `indexserverconfig.xml`.

- a. Set the sub-index quantity thresholds:

- `lmpi-finalmerge-threshold-cleanentry-mustmerge`

If the number of sub-indexes on the collection reaches this value, makes it a high priority (must-merge) to run a final merge. Default: 10.

- `lmpi-finalmerge-threshold-cleanentry-notmerge`

If the number of sub-indexes on the collection falls below this value, makes it a low priority (do-not-merge) to run a final merge. Default: 2.

Set these two properties in conjunction with the `xdb.lucene.nonFinalMaxMergeSize` property in `xdb.properties`, which the scheduler uses to calculate the number of sub-indexes to merge. The larger the value of `xdb.lucene.nonFinalMaxMergeSize`, the smaller the value of `xdb.lucene.nonFinalMaxMergeSize`, and vice versa.

Only sub-indexes with size above `xdb.lucene.nonFinalMaxMergeSize` are considered for final merge by the threshold-based scheduler. Smaller sub-indexes are handled by the more frequent non-final merge process.

- b. Set the black node quantity thresholds:

- `lmpi-finalmerge-threshold-bl-mustmerge`

If the number of black nodes on a collection is greater than this value, makes it a high priority (must-merge) to run a final merge. Default: 50000.

- `lmpi-finalmerge-threshold-bl-notmerge`

If the number of black nodes on the collection is smaller than this value, makes it a low priority (do-not-merge) to run a final merge. Default: 5000.

For information about final merge priorities and prioritization rules, see [“Final merging priorities and prioritization rules” on page 381](#).

4. Set the schedule for final merges of different priority levels:

- `lmpi-finalmerge-threshold-interval`

Specify the time interval in seconds at which to run medium priority (optional-merge) final merges. Default: 14400 (4 hours).



**Note:** This interval setting is only effective in threshold-based scheduling. Do not confuse this with the interval settings in the interval-based and Cron-style schedulers.

- `lmpi-finalmerge-oap-weekend`

Specify weekend days (24 hours a day) on which to run medium priority (optional-merge) final merges. Specify numbers that correspond to system weekdays (not calendar weekdays): 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday. Delimit multiple days with comma (,); for example:

```
lmpi-finalmerge-oap-weekend=1,7
```

This setting overrides the time slot settings specified in the `lmpi-finalmerge-oap-day` property.

- `lmpi-finalmerge-oap-day`

Specify in the `hh:mm` format time slots (`<StartTime>-<EndTime>`) in a day in which to run medium priority (optional-merge) final merges. Multiple time slots cannot overlap; Delimit them with comma (,); For example:

```
lmpi-finalmerge-oap-day=20:15-8:15,12:01-13:30
```

- `lmpi-finalmerge-parallel-execution-crossnode`

Specify whether to allow multiple final merges on different nodes to run simultaneously. Set this to false if multiple nodes share the same storage area to prevent I/O bottleneck. Default: true.

5. Restart xPlore for the configurations to take effect.

## 15.8.5 Setting final merge blackout periods

The final merge is very CPU-intensive and I/O-intensive and may cause noticeable drop in both indexing and query performance. To prevent scheduled final merges from running on an xPlore instance during performance-critical periods such as business hours, you can set final merge blackout periods.

- If you use the interval-based or Cron-style scheduler, set the final merge blackout period at the xDB level:

Edit the `xdb.properties` file under `\deploy\dsearch.war\WEB-INF\classes` of the instance, set the `finalMergingBlackout` property in the following format:

```
xdb.lucene.finalMergingBlackout = <StartHour>-<EndHour>
```

For example, if you set the value to 8-20, the blackout period will be from 8 a.m. to 8 p.m. every day, and scheduled final merges will not start to run during this period.

Using this method, you can only set one blackout period.

Once you have set the `finalMergingBlackout` property in `xdb.properties` file, this property overrides the Interval-based scheduling and Cron-style scheduling.

- If you use the threshold-based scheduler, set the final merge blackout period at the xPlore level:

In `indexserverconfig.xml`, set the following property:

```
lmpi-finalmerge-day-blackout = <StartTime>1-<EndTime>1, <StartTime>2-<EndTime>2, ...
```

Using this method, you can set one or more blackout periods in a day. Delimit multiple time slots with comma (,), for example: `lmpi-finalmerge-day-blackout = 8:15-15:00, 16:15-20:00` Time slots cannot overlap.

Except for manually triggered final merges, no final merges, regardless of their priority, can be triggered during blackout periods.

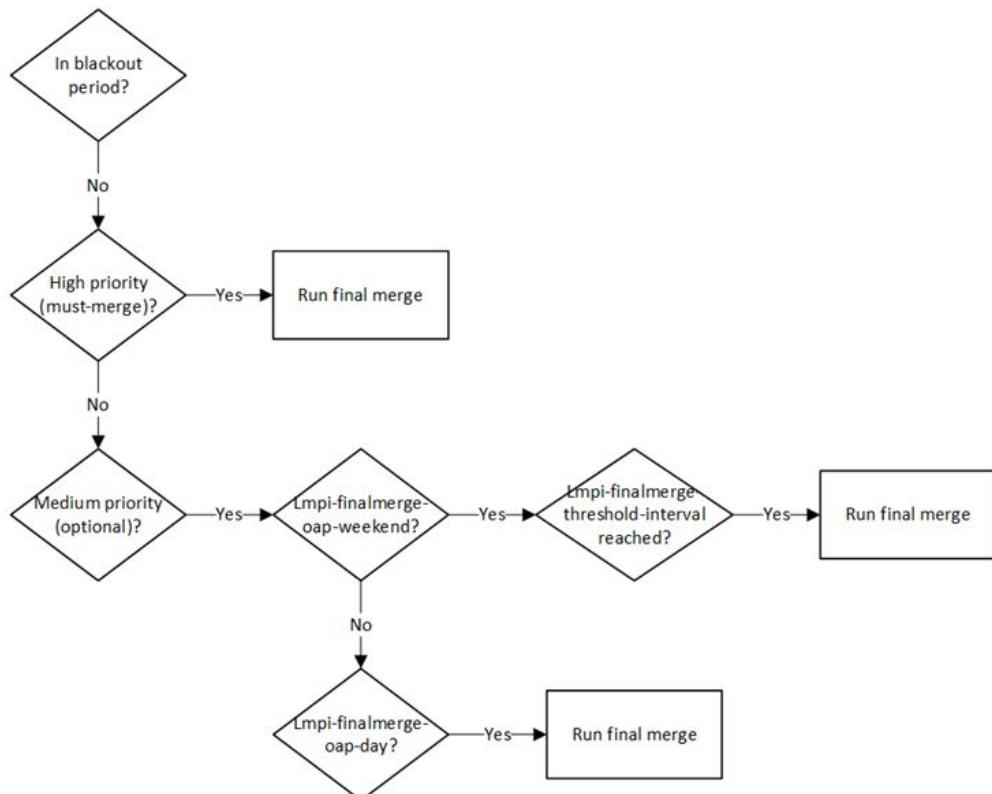
As with other scheduler settings, when the threshold-based scheduler is enabled, the `lmpi-finalmerge-day-blackout` property overrides the `xdb.lucene.finalMergingBlackout` property.

Blackout periods do not stop an already running final merge process. A scheduled final merge started before the blackout start hour will continue to run into or even past defined final merge blackout periods without being affected. If running final-merges out of blackout degrades the system performance, take buffer into account when configuring blackout periods. Also, manually started final merges ignore any blackout periods.

### 15.8.6 How threshold-based scheduling works

The threshold-based scheduler run final merges based on the following rules:

- Under any circumstances, never start final merges within the blackout periods.
- At any point in time, only one final merge can run on a single node.
- Start final merges only when necessary and in order of priority (from high to low)
  - Run high priority (must-merge) final merges first
  - Run medium priority (optional-merge) final merges only at specified intervals (`lmpi-finalmerge-threshold-interval`) during the weekend (`lmpi-finalmerge-oap-weekend`) and within specified time slots each workday (`lmpi-finalmerge-oap-day`).
  - Low priority (not-merge) final merges will not be run under any circumstances.
  - If there are multiple eligible final merges, the one with more sub-indexes takes precedence.



## 15.9 Documentum index agent performance

### Index agent settings

The parameters described in this section can affect index agent performance. Do not change these values unless you are directed to change them by OpenText Global Technical Services.

In migration mode, set the parameters in the `indexagent.xml` located in `<index_agent_WAR>/WEB-INF/classes/`.

In normal mode, also set the corresponding parameters in the `dm_ftindex_agent_config` object. In normal mode, index agent configuration is loaded from `indexagent.xml` and from the `dm_ftindex_agent_config` object. If there is a conflict, the settings in the config object override the settings in `indexagent.xml`.

- `exporter.thread_count` (`indexagent.xml`) / `exporter_thread_count` (`dm_ftindex_agent_config`)  
Number of threads that extract metadata into dftxml using DFC.
- `connectors.file_connector.batch_size` (`indexagent.xml`) / `connectors_batch_size` (`dm_ftindex_agent_config`)  
Number of items picked up for indexing when the index agent queries the repository for queue items.
- `exporter.queue_size` (`indexagent.xml`) / `exporter_queue_threshold` (`dm_ftindex_agent_config`)  
Internal queue of objects submitted for indexing.
- `indexer.queue_size` (`indexagent.xml`) / `indexer_queue_threshold` (`dm_ftindex_agent_config`)  
Queue of objects submitted for indexing.
- `indexer.callback_queue_size` (only in `indexagent.xml`, used for both migration and normal mode)  
Size of queue to hold requests sent to xPlore for indexing. When the queue reaches this size, the index agent waits until the callback queue has reached 100% less the `callback_queue_low_percent`.

### Measuring index agent performance

Verify index agent performance using the index agent UI details page. Find the details for *Indexed content KB/sec* and *Indexed documents/sec*. *All Averages* measures the average time between index agent startup and current run time. *All Averages up to Last Activity* measures the time between index agent startup and last indexing activity.

## 15.10 Indexing performance

Various factors affect the rate of indexing. You can tune some indexing and xDB parameters and adjust allowable document size.

### Factors in indexing rate

The following factors affect indexing rate:

- The complexity of documents

For example, a simple text document containing thousands of words can take longer to index than a much larger Microsoft Word document full of pictures. MS Excel files take much longer to index due to their complex cell structure.

- The indexing server I/O subsystem capabilities
- The number of CPS instances

For heavy ingestion loads or high availability requirements, add CPS instances to increase content processing bandwidth.

- The number of collections

Create multiple collections spread over multiple xPlore instances to scale xPlore. (Documents can be indexed into specific target collections. For best search performance, route queries to specific collections. See “[Routing a query to a specific collection](#)” on page 317)

- Recovery during heavy ingestion

If the system crashes during a period of heavy ingestion, transactional recovery could take a long time as it replays the log. The recovery process is single-threaded. Alternatively, you can set up an active/active high availability system so that failure in a single system does not disrupt business.

### Creating temporary collections for ingestion

You can create a collection and ingest documents to that collection. After ingestion has completed, move the collection to become a subcollection of existing collection. See “[Moving a collection](#)” on page 202.

### Tunable indexing properties

The number of threads, batch size, tracking DB cache size, thread wait time, and queue size at each stage of indexing impacts ingestion performance. The biggest impact on ingestion rate is with threadpool size and processing buffer size. You can configure CPS and indexing settings using xPlore administrator. For a list of these properties, see “[Document processing and indexing service configuration parameters](#)” on page 402.

To scale up for large ingestion requirements or for high availability, add more CPS instances.

## Document size and performance

Several configuration properties affect the size of documents that are indexed and consequently the ingestion performance. “[Maximum document and text size](#)” on page 124 describes these settings.

### Tuning xDB properties for indexing performance

Most applications do not need to modify xDB properties. The bottleneck for indexing is usually the process of writing index files to disk, which you can address by increasing I/O capabilities. Indexing can be slowed by the merge process in which additions or modifications to the index are merged into xDB. With the guidance of OpenText Global Technical Services, you can set merge scheduling or interrupt a merge. See “[Managing sub-index merges](#)” on page 377.

You can tune the following property for indexing in xdb.properties, which is located in the directory WEB-INF/classes of the primary instance. If this property is not listed, you can add it.

- *xdb.lucene.ramBufferSizeMB*: Size in megabytes of the RAM buffer for document additions, updates, and deletions. For faster indexing, use as large a RAM buffer as possible for the host. Default: 30.

## 15.11 Search performance

### Changing the security cache sizes

Monitor the query audit record to determine security performance. The value of <TOTAL\_INPUT\_HITS\_TO\_FILTER> records how many hits a query had before security filtering. The value of <HITS\_FILTERED\_OUT> shows how many hits were discarded because the user did not have permissions for the results. The hits filtered out divided by the total number of hits is the hit ratio. A low hit ratio indicates an underprivileged user, who often has slower query response times than other users.

You can improve security filter performance by adjusting values of security filter properties. For information on how to change these configuration settings, see “[Configuring the security cache](#)” on page 67

If you have many ACLs, increase the value of *global-ace-cache-size*, *acl-cache-size* and *global-acl-cache-user-count*.

### Increasing query batch size

In a Documentum client application based on DFC, you can set the query batch size. Edit dfc.properties on the search client to increase the value of *dfc.batch\_hint\_size*. Default: 50. Suggested size: 350.

- “[Troubleshooting slow queries](#)” on page 305
- “[Measuring performance](#)” on page 374

## 15.11.1 About search performance

### Factors in query performance

The following features of full-text search can affect search performance:

- Single-box search in Webtop

The default operator for multiple terms is AND. The user can explicitly combine terms with OR (the old Webtop default), but performance is much slower. A DFC customization of Webtop search or a DFC filter can also change the default from AND to OR.

- Flexible metadata search (FTDQL)

Searches on multiple object attributes can affect performance, especially when the first term is unselective.

- Leading or trailing wildcards

By default, xPlore does not match parts of words. For example, *WHERE object\_name LIKE 'foo%'* matches *foo bar* but not *football*. Support for fragment matches (leading and trailing wildcards) can be enabled, but this impacts performance. A more limited support for leading wildcards in metadata search can also be enabled.

- Security

Native xPlore security performs faster than security applied to results in the Documentum Server. The latter option can be enabled, but this impacts performance.

- Number of documents

Documents can be routed to specific collections based on age or other criteria. When queries are routed to a collection, performance is much better. Scaling to more instances on the same or multiple hosts, as well as use of 64-bit hosts, can also improve search performance.

- Size of query result set

Consume results in a paged display for good performance. Webtop limits results to 350, with a smaller page size (from 10 to 100). The first page of results loads while the remaining results are fetched. CenterStage limits results to 150. Paging is especially important to limit result sets for underprivileged users.

- Number of collections

If queries are not run in parallel mode (across several collections at once), response time rises as the number of collections rises. Queries can be targeted to specific collections to avoid this problem. If you do not use targeted queries, try to limit the number of collections in your xPlore federation. For information on targeted queries, see “[Routing a query to a specific collection](#)” on page 317. To set parallel mode for DFC-based search applications, set the following property in `dfc.properties` to true:

```
dfc.search.xquery.option.parallel_execution.enable = true
```

- Caches empty on system startup

At startup, the query and security caches have not been filled, so response times are slower. Make sure that you have allocated sufficient memory for the file system buffer cache and good response time from the I/O subsystem.

- Response times slower during heavy ingestion

Slow queries during ingestion are usually an issue only during migration from FAST. If your environment has large batch migrations once a month or quarterly, you can set the target collection or domain to index-only during ingestion.

Alternatively, you can schedule ingestion during an off-peak time.

## Measuring query performance

Make sure that search auditing is enabled.

Examine the query load to see if the system is overloaded. Run the report **Top N slowest queries**. Examine the *Start time* column to see whether slow queries occur at a certain period during the day or certain days of the month.

Save the query execution plan to find out whether you need an additional index on a metadata element. (For more information on the query plan, see “[Debugging queries](#)” on page 311.) Documentum clients can save the plan with the following iAPI command:

```
apply,c,NULL,MODIFY_TRACE,SUBSYSTEM,S,fulltext,VALUE,S,ftengine
```

The query execution plan recorded is in dsearch.log, which is located in the logs subdirectory of the WildFly deployment directory.

## 15.11.2 Tuning CPS and xDB for search

### Tuning CPS for query performance

The following parameters for high-volume ingestion environments can improve query response. Configure the following parameters in the CPS configuration file PrimaryDsearch\_local\_configuration.xml, which is located in the CPS instance directory *xplore\_home/dsearch/cps/cps\_daemon*. If these properties are not in the file, you can add them. These settings apply to all CPS instances.

- `add_failure_documents Automatically`: Specifies whether to add automatically documents that fail processing to the failed document ID list stored in the file designated by the `failure_document_id_file` parameter. Default: false. If you turn on this feature, make sure the `failure_document_id_file` is exclusively used by the current CPS instance and not shared by other CPS instances.
- `connection_pool_size`: Specifies how many connections CPS manager allocates to each daemon. If CPU and memory allow, add to `daemon_count`, then increase `connection_pool_size` to use a reasonable amount of memory. Set this value to 3 or greater.
- `cut_off_text`: Set to true to cut off text of large documents that exceed `max_text_threshold` instead of rejecting entire document. Default: false.

Documents that are partially indexed are recorded in `cps_daemon.log`: *docxxxx is partially processed*. The dftxml is annotated with the element `partiallyIndexed`.

- `daemon_count`: Specifies the number of daemons that handle normal indexing and query requests (not a dedicated query daemon). Set from 1 to 6. Default: 1. For information about adding CPS daemons, see “[Adding CPS daemons for ingestion or query processing](#)” on page 146.
- `daemon_restart_threshold`: Specifies how many requests a CPS daemon can handle before it restarts. Default: 1000.
- `daemon_restart_memory_threshold`: Set a value in bytes for maximum CPS memory consumption. After this limit, CPS will restart. A maximum of 8 GB is recommended. Default: 4000000000.
- `daemon_restart_consistently`: Specifies whether CPS should restart regularly after it is idle for 5 minutes. Default: true.
- `dump_context_if_exception`: Specifies whether to dump stack trace if exception occurs. Default: true.
- `failure_document_id_file`: The file that contains IDs of failed documents to be skipped. You can edit this file. IDs of failed documents are added to it automatically if `add_failure_documents_automatically` is set to true. Default: `<xplore_home>/cps/skip_failure_document.txt`.
- `io_block_unit`: Logical block unit of the read/write target device. Default: 4096.
- `io_chunk_size`: Size for each read/write chunk. Default: 4096.
- `linguistic_processing_time_out`: Interval in seconds after which a CPS hang in linguistic processing forces a restart. Valid values: 60 to 360. Default: 360.
- `load_content_directly`: For internal use only.
- `query_dedicated_daemon_count`: The number of CPS daemons dedicated to query processing. Other CPS daemons handle ingestion when there is a dedicated query daemon. Valid values: 0 to 3. Default: 1.
- `retry_failure_in_separate_daemon`: Specifies whether to retry failed documents in a newly spawned CPS daemon. Default: true. A retry daemon is not limited by the value of `daemon_count`.
- `skip_failure_documents`: Specifies whether CPS should skip documents that fail processing instead of retrying them, to reduce CPS crashes. Default: true. Failed documents are retried once unless this property is set to true.
- `skip_failure_documents_upper_bound`: Specifies the maximum number of failed documents that CPS will record in the failure document. Valid values: integers. Default: -1 (no upper bound)
- `text_extraction_time_out`: Interval in seconds after which a CPS hang in text extraction forces a restart. Valid values: 60 to 300. Default: 300.
- `use_direct_io`: Requires CPS to read and write staging files to devices directly. Default: false. If most incoming files are local, use the default caching. If most files are remote, use direct IO.

## Tuning xDB properties for search performance

Search performance can be slowed during index merges. For information on scheduling and interrupting merges, see “[Managing sub-index merges](#)” on page 377.

You can also limit query results set size, which is 12000 results by default. This default value supports facets. If your client application does not support facets, you can lower the result set size. Open `xdb.properties`, which is located in the directory `WEB-INF/classes` of the specified xPlore instance. Set the value of `queryResultsWindowSize` to a number smaller than 12000.

### 15.11.3 Creating a CPS daemon dedicated to search

Queries are processed in CPS linguistic analysis. During periods of high ingestion, queries can be slow. You can configure CPS instances for search only. You can configure a processing daemon within a CPS instance that processes only queries.

To configure a CPS instance for search only, see “[Configuring CPS dedicated to indexing or search](#)” on page 122.

To configure a CPS daemon dedicated to search, edit the CPS configuration file `PrimaryDsearch_local_configuration.xml`, which is located in the CPS instance directory `xplore_home/dsearch/cps/cps_daemon`. Change the following properties, or add them if they are not present:

- `daemon_count`: Sets the number of daemons that process indexing requests. Set from 1 to 8. Default: 1. If `query_dedicated_daemon_count` is greater than 0, other daemons process only indexing requests.
- `query_dedicated_daemon_count`: Sets the number of CPS daemons dedicated to query processing (no ingestion). Valid values: 0 to 3. Default: 1. Set to 1 and increase `daemon_count` to 2 or more.



**Note:** For local CPS, the total daemon count (`daemon_count` and `query_dedicated_daemon_count` put together) should be less than 8. Consider a case in which `daemon_count` is 6 and `query_dedicated_daemon_count` is 1, you may still face CPS processing bottleneck. For such a scenario, you should consider adding remote CPS on other hosts. For remote CPS, there is no limitation for the total daemon count.

See also: “[Tuning CPS and xDB for search](#)” on page 391.

### 15.11.4 Improving search performance with time-based collections

You can plan for time-based collections, so that only recent documents are indexed. If most of your documents are not changed after a specific time period, you can migrate data to collections. Base your migration on creation date, modification date, or a custom date attribute.

Route using a custom routing class, index agent configuration, or customized DFC query builder. You can migrate a limited set of documents using time-based DQL in the index agent UI.

To determine whether a high percentage of your documents is not touched after a specific time period, use two DQL queries to compare results:

1. Use the following DQL query to determine the number of documents modified and accessed in the past two years (change DQL to meet your requirements):

```
select count(*) from dm_sysobject where  
datediff(year,r_creation_date,r_access_date)<2 and  
datediff(year,r_creation_date,r_modify_date)<2
```

2. Use the following DQL query to determine the number of documents in the repository:

```
? ,c,select count(*) from dm_sysobject
```

3. Divide the results of step 1 by the results of step 2. If the number is high, for example, 0.8, most documents were modified and accessed in the past two years. (80%, in this example)

## 15.12 Throttling indexing and searching

xPlore is an enterprise full text search engine which serves indexing and searching requests from outside, meanwhile scheduling merging and other activities from inside. These activities require high usage of CPU, memory, and storage IO. If there is no mechanism to manage resource allocation wisely, the resource-consuming activities could lead to an unrecoverable state such as OOM.

To maintain the system in a healthy state while processing various jobs, you can use the throttling feature to regulate the index speed and search concurrency dynamically.

### Throttling indexing

Index speed is throttled when either a final merge is running or the number of search queries xPlore is executing (also known as active search count) exceeds a certain threshold.

Index speed is defined as the content size that the index service processes per second. The content size is the sum of the dftxml size and the value of the *r\_content\_size* attribute. For delete operations, as the index request does not contain dftxml, the system uses a constant value 4k as the content size.

The active search count stands for the number of search queries xPlore is executing at the current time point, which increases by one each time a search request is submitted to xPlore and decreases by one each time a search request completes.

By default, index throttling is enabled. To adjust the speed limit of index throttling, modify the following property under `node->properties` in `indexserverconfig.xml`.

*index-throttle-speed-kbps-limit*: Specifies index speed in kilobyte per second. Default value: 400. To turn off index throttling, set this value to -1.

Final merges cost a lot storage I/O and disk usage. Index speed is throttled to the value specified in *index-throttle-speed-kbps-limit* when a final merge starts. The speed is restored when the final merge ends.

When the system is executing search queries, it is often desirable to throttle index to free more resource for search. In addition to *index-throttle-speed-kbps-limit*, set the following property under `node->properties` in `indexserverconfig.xml`.

*index-throttle-active-search-threshold*: The threshold of the active search count. Index speed is throttled when the active search count exceeds this threshold. Default value: 2

The Index speed is restored when the active search count decreases to a value lower than the threshold.

## Throttling searching

The search service is throttled via limiting the number of concurrent search query jobs when GC busyness is higher than a certain value.

GC busyness is measured as a floating number ranging in between [0, 1]. The number represents the portion of time JVM's garbage collection takes in the last 60 seconds. For example, 0.5 indicates that JVM spent a total of 30 seconds on garbage collection in the last 60 seconds.

By default, search throttling is enabled. To adjust the search throttling behavior, modify the following properties under `node->properties` in `indexserverconfig.xml`.

- *throttle\_gc\_busyness\_limit*: The GC overhead threshold. If xPlore GC overhead exceeds this threshold, the number of concurrent search query jobs is limited. Default value: 0.4.
- *throttle\_control\_query\_concurrency\_limit*: The maximum number of concurrently running search query jobs allowed during search throttling. Default value: 2. To turn off index throttling, set this value to -1.

Search throttling ends when the GC overhead decreases to a value lower than *throttle\_gc\_busyness\_limit*.



# Appendix A. Index Agent, CPS, Indexing, and Search Parameters

## A.1 dm\_ftengine\_config

### Attributes

Following are attributes specific to dm\_ftengine\_config. Some attribute values are set by the index agent when it creates the dm\_ftengine\_config object.

 **Note:** Not all attributes values are set at object creation. If you do not set values, the default values are used. For instructions on changing the attribute values, see “[Query plugin configuration \(dm\\_ftengine\\_config\)](#)” on page 273.

For iAPI syntax to change attributes, see “[Query plugin configuration \(dm\\_ftengine\\_config\)](#)” on page 273.

Attribute	Description
acl_check_db:	If ftsearch_security_mode is set to xPlore and this value is set to true, xPlore filters the results based on its ACL information, and then Documentum Server filters again based on current database ACL information from the filtered results set (double security check). You must disable XQuery generation. See “ <a href="#">Changing search results security</a> ” on page 63.
acl_domain:	Owner of dm_fulltext_admin_acl (user specified at installation)
acl_name	dm_fulltext_admin_acl
default_fuzzy_search_similarity	Controls the degree of similarity between a search term and an index term. Default; 0.5. See “ <a href="#">Configuring fuzzy search</a> ” on page 264.
dsearch_config_host	Specifies the fully qualified host name or IP address of the xPlore host that the index agent connects to.
dsearch_config_port	Specifies the HTTP or HTTPS port that the index agent connects to.
dsearch_domain	Name of repository
dsearch_override_locale	Overrides the locale of the query with the specified locale.
dsearch_qrserver_host	Specifies the fully qualified host name or IP address of the xPlore host that the Documentum Server query plugin connects to.

Attribute	Description
dsearch_qrserver_port	Specifies the HTTP or HTTPS port of the xPlore host that the Documentum Server query plugin connects to.
dsearch_qrserver_protocol	Sets HTTPS or HTTP as connection protocol.
dsearch_qrserver_target	xPlore index server servlet: Partial URL combined with host, protocol and port to create full URL.
dsearch_qrygen_mode	For internal use only.
dsearch_result_batch_size	Sets the number of results fetched from xPlore in each batch. Default: 200.
fast_wildcard_compatible (replaces fds_contain_fragment)	Sets fragment search option. Default: false.
filter_config_id	Most recent object ID of dm_filter_config type
folder_cache_limit	Specifies the maximum number of maximum folder IDs included in the index probe. Default: 2000. If folder descend condition evaluates to less than folder_cache_limit value, then folder ids are pushed into index probe. Otherwise, the folder constraint is evaluated separately for each result. Raise the value if folder descend queries are slow or timing out. Lower the value if folder descend queries cause out of memory to too many clauses errors. See “Query plugin configuration (dm_ftengine_config)” on page 273.
ft_collection_id	Repeating attribute that references a collection object of the type dm_fulltext_collection. Reserved for use by Documentum Server client applications.
ft_wildcards_mode	Specifies how wildcards are evaluated in full-text clauses (Webtop simple search, Webtop advanced search Contains field, and global search in xCP 2.0). Valid values: <ul style="list-style-type: none"> <li>• none: Wildcard is treated as a literal * character.</li> <li>• explicit: Wildcard character must be entered to search for fragments (default).</li> <li>• implicit: Wildcards are added implicitly around every search term (negative impact on performance).</li> <li>• trailing_implicit: A wildcard is added to the end of every search term.</li> </ul>

Attribute	Description
ftsearch_security_mode	0: Documentum Server. DFC search service will not use IDfXQuery and instead will generate DQL. 1: xPlore (default)
fuzzy_search_enable	Specifies whether fuzzy search is applied. Default: false. See <a href="#">"Configuring fuzzy search" on page 264</a> .
group_name	dm_fulltext_admin
object_name	Dsearch Fulltext Engine Configuration FAST
query_plugin_mapping_file	Path on Documentum Server host to mapping file. This file maps attribute conditions to the XQuery subpaths.
query_timeout	Sets the interval in milliseconds for a query to time out. Default: 60000.
security_mode	Sets summary security mode. See <a href="#">"Configuring summary security" on page 263</a> .
thesaurus_search_enable	Set to true to enable thesaurus search.
use_thesaurus_on_phrase	Set to true to match entire phrases in the thesaurus.
parallel_summary_computing_enable	Set to true to enable parallel summary calculation and automatically convert non-parallel summary queries into parallel summary queries in query generation.

## A.2 Index agent configuration parameters

### General index agent parameters

The index agent configuration file `indexagent.xml` is located in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`. Most of these parameters are set at optimal settings for all environments.

**Table A-1: Indexagent configuration parameters in generic\_indexer.parameter\_list**

Parameter name	Description
acl_exclusion_list	Add this parameter to exclude specific ACL attributes from indexing. Contains an <i>acl_attributes_exclude_list</i> element. Check with OpenText Global Technical Services before you add or modify this list.
acl_attributes_exclude_list	Specifies a space delimited list of ACL attributes that will not be indexed.
collection	Specifies the name of a collection or a list of connections separated with comma to which all documents will be routed. Default: null, meaning that xPlore routes documents using the default rule.
dsearch_qrserver_host	Fully qualified host name or IP address of host for xPlore server
dsearch_qrserver_port	Port used by xPlore server. Default is 9200
dsearch_domain	Repository name
group_exclusion_list	Add this parameter to exclude specific group attributes from indexing. Contains a <i>group_attributes_exclude_list</i> element. Check with OpenText Global Technical Services before you add or modify this list.
group_attributes_exclude_list	Specifies a space delimited list of group attributes that will not be indexed.
index_type_mode	Object types to be indexed. Values: both (default)   aclgroup   sysobject. If you use two index agents, each can index either ACLs or sysobjects.
max_requests_in_batch	Maximum number of objects to be indexed in a batch. Default: 10
max_batch_wait_msec.	Maximum wait time in milliseconds for a batch to reach the <i>max_requests_in_batch</i> size. When this timeout is reached, the batch is submitted to xPlore. The default setting (1) is for high indexing throughput. If your index agent has a low ingestion rate of documents and you want to have low latency, reduce both <i>max_requests_in_batch</i> and <i>max_submission_timeout_sec</i> .
max_pending_requests	Maximum number of indexing requests in the queue. Default: 10000

Parameter name	Description
max_tries	Maximum number of tries to add the request to the internal queue when the queue is full. Default: 2

## General index agent runtime parameters

Requests for indexing pass from the exporter queue to the indexer queue to the callback queue.

**Table A-2: Index agent runtime configuration in indexer element**

Parameter	Description
queue_size	Size of queue for indexing requests. When the queue reaches this limit, the index agent will wait for the queue to be lower than queue_size less (queue_size * queue_low_percent). For example, if the queue_size is 500 and queue_low_percent is 10%, then the agent will resume indexing when the queue is lower than 500 - (500 * .1) = 450.
queue_low_percent	Percent of queue size at which the index agent will resume processing the queue.
callback_queue_size	Size of queue to hold requests sent to xPlore for indexing. When the queue reaches this size, the index agent will wait until the callback queue has reached 100% less the callback_queue_low_percent.
callback_queue_low_percent	Percent of callback queue size at which the index agent will resume sending requests to xPlore.
callback_queue_wait_for_result_interval	Interval in milliseconds the IndexAgent will wait if an item is not in the callback queue.
callback_queue_wait_for_result_times	Number of times the IndexAgent waits and checks for an item in the callback queue.
wait_time	Time in seconds that the indexing thread waits before reading the next item in the indexing queue.
thread_count	Number of threads to be used by index agent.
shutdown_timeout	Time the index agent should wait for thread termination and cleanup before shutdown.
runaway_timeout	Timeout for runaway requests..

Parameter	Description
content_clean_interval	Timeout to clean local content area. Set to the same value as runaway timeout. After the local content area is cleaned, documents may still remain in the queue for xPlore processing.
partition_config	You can add this element and its contents to map partitions to specific collections. See “ <a href="#">Mapping Server storage areas to collections</a> ” on page 90.

## Other index agent parameters

**Table A-3: Other index agent parameters**

Parameter	Description
contentSizeLimit	In exporter.parameter_list. Sets the maximum size for documents to be sent for indexing. The value is in bytes. Default: 20MB.

## A.3 Document processing and indexing service configuration parameters

You can configure the following settings for the CPS and indexing services in xPlore administrator. For CPS and indexing processing settings, select **System Overview** and select **Global Configuration**; > **Index Service**. The default values are optimized for most environments.

### Document processing (CPS) global parameters

The per-instance CPS settings are for the instance and do not overlap with the CPS settings in Indexing Service configuration.

- CPS-threadpool-size: Number of threads for the CPS thread pool. Default: number of OS processors.
- CPS-requests-batch-size: Maximum number of CPS requests in a batch. Default: 5.
- CPS-thread-wait-time: Time in milliseconds to accumulate requests in a batch. Range: 1-2147483647. Default: 1000.
- CPS-executor-queue-size: Maximum size of CPS queue before spawning a new worker thread. Default: 10.
- query\_daemon\_restart\_threshold: Maximum number of requests one query daemon can process.

## Indexing global parameters

- index-requests-max-size: Maximum size of internal index queue. Default: 1000.
- index-requests-batch-size: Maximum number of index requests in a batch. Default: 10.
- index-threadpool-size: Number of threads for the index thread pool. Default: 2.
- index-thread-wait-time: Maximum wait time in milliseconds to accumulate requests in a batch. Range: 1-2147483647. Default: 1000.
- index-executor-queue-size: Maximum size of index queue before spawning a new worker thread. Default: 10.
- index-check-duplicate-at-ingestion: Set to true to check for duplicate documents. May slow ingestion. Default: true.
- enable-subcollection-ftindex: Set to true to create a multi-path index to search on specific subcollections. Ingestion is slower, especially when you have multiple layers of subcollections. If false, subcollection indexes are not rebuilt when you rebuild a collection index. Default: false.
- index-overwrite-same-version: if set to true, an indexed dftxml file will be overwritten by a new dftxml file although their version is the same. If false, the new dftxml file will be skipped. Default: true.
- index-contents-consistently: If set to true, and a file's content cannot be indexed, then its metadata will not be indexed . If false, only the metadata will be indexed. Default: false.
- rebuild-index-batch-size: Sets the number of documents to be reindexed. Default: 1000.
- rebuild-index-embed-content-limit: Sets the maximum size of embedded content for language detection in index rebuild. Larger content is streamed. Default: 2048.
- commit-option: Default: -1.
- rebuild-index-batch-size: Number of documents to add to rebuild the index in a batch. Default: 1000.
- rebuild-index-thread-number: Number of threads used to rebuild a collection. Default: 1 (parallel rebuild mode is off).
- rebuild-index-embed-content-limit: Maximum embedded content for index rebuilding. Larger content is passed in a file, not embedded. Default: 2048.

## CPS instance configuration parameters

You can configure the following CPS settings for each instance in xPlore Administrator. The values are recorded in the file `<instance_name>_local_configuration.xml`, where `<instance_name>` is the name of the xPlore instance you are configuring CPS settings for; for example, PrimaryDsearch. This file is located in `xplore_home/dsearch/cps/cps_daemon`. The default values have been optimized for most environments.

- Connection pool size: Maximum number of concurrent connections. Valid values: 1-100. Default: 4.  
Increasing the number of connections consumes more memory. Decreasing can slow ingestion.
- Port number: Listener port for CPS daemon, used by the CPS manager. Default: 9322.  
This value is set during xPlore configuration.
- Daemon path: Specifies the path to the installed CPS daemon (read-only).  
This value is set during xPlore configuration.
- Keep intermediate temp file: Keep content in a temporary CPS folder for debugging.  
Enabling temp file has a large impact on performance. Disable (default) to remove temporary files after the specified time in seconds. Time range in seconds: 1-604800 (1 week).
- Restart threshold: Select **After processed...** and specify the number of requests after which to restart the CPS daemon.  
Disable if you do not want the daemon restarted. Decreasing the number can affect performance.
- Heartbeat: Interval in seconds between the CPS manager and daemon.  
Range: 1-600. Default: 60.
- Embedded return: Select **Yes** (default) to return embedded results to the buffer. Check **No** return results to a file, and specify the file path for export.  
Embedded return increases communication time and impacts ingestion.
- Export file path: Valid URI at which to store CPS processing results, for example, file:///c:/.  
If the results are larger than *Result buffer threshold*, they are saved in this path. This setting does not apply to remote CPS instances, because the processing results are always embedded in the return to xPlore.
- Result buffer size threshold: Number of bytes at which the result buffer returns results to file.  
Valid values: 8 - 16 MB. Default: 1 MB (1048576 bytes). Larger values can accelerate the process but can cause more instability.
- Processing buffer size threshold: Specifies the number of bytes of the internal memory chunk used to process small documents.  
If this threshold is exceeded, a temporary file is created for processing. Valid values: 100 KB-10 MB. Default: 2 MB (2097152 bytes). Increase the value to speed processing. This consumes more memory.
- Load file to memory: Check to load the submitted file into memory for processing. Uncheck to pass the file to a plug-in analyzer for processing (for example, the Documentum index agent).

- Batch in batch count: Average number of batch requests in a batch request.  
Range: 1-100. Default: 5. CPS assigns the number of Connection pool threads for each batch\_in\_batch count. For example, defaults of batch\_in\_batch of 5 and connection\_pool\_size of 5 result in 25 threads.
- Thread pool size: Number of threads used to process a single incoming request such as text extraction and linguistic processing.  
Range: 1-100. Default: 8). Larger size can speed ingestion when CPU is not under heavy load. This causes instability at heavy CPU load.
- System language: ISO 639-1 language code that specifies the language for CPS.
- Max text threshold: Sets the size limit, in bytes, for the text within documents.  
Range: 5MB - 2GB in bytes. Default: 10485760 (10 MB). Maximum setting: 2 GB. Larger values can slow ingestion rate and cause more instability. Above this size, only the document metadata is tokenized.  
Includes expanded attachments. For example, if an email has a zip attachment, the zip file is expanded to evaluate document size. If you increase this threshold, ingestion performance can degrade under heavy load.
- Illegal char file: Specifies the URI of a file that defines illegal characters.  
To create a token separator, xPlore replaces illegal characters with white space. This list is configurable.
- Request time out: Number of seconds before a single request times out.  
Range: 60-3600. Default: 600.
- Daemon standalone: Check to stop daemon if no manager connects to it. Default: false.
- IP version: Internet Protocol version of the host machine. Values: IPv4 or IPv6. Dual stack is not supported.



**Note:** For CPS to be started successfully, ensure that you set the correct IP version before starting xPlore.

- Use express queue: This queue processes query requests. Queries are processed for language identification, lemmatization, and tokenization. The express queue has priority over the regular queue. Set the maximum number of requests in the queue. Default: 128.
- The regular queue processes indexing requests. Set the maximum number of requests in the queue. Default: 1024.
- When the token count is zero and the extracted text is larger than the configured threshold, a warning is logged

You can configure the following additional parameters in the CPS configuration file PrimaryDsearch\_local\_configuration.xml, which is located in the CPS instance directory `xplore_home/dsearch/cps/cps_daemon`. If these properties are not in the file, you can add them.

- *detect\_data\_len*: The number of bytes used for language identification. The bytes are analyzed from the beginning of the file. A larger number slows the ingestion process. A smaller number increases the risk of language misidentification. Default: 65536.
- *max\_batch\_size*: Limit for the number of requests in a batch. Valid values: 2 - 65535 (default: 65535).



**Note:** The index agent also has batch size parameters.

- *max\_data\_per\_process*: The upper limit in bytes for a batch of documents in CPS processing. Default: 30 MB. Maximum setting: 2 GB.
- *normalize\_form*: Set to true to remove accents in the index, which allows search for the same word without the accent.
- *slim\_buffer\_size\_threshold*: Sets memory buffer for CPS temporary files. Increase to 16384 or larger for CenterStage or other client applications that have a high volume of metadata.
- *temp\_directory*: Directory for CPS temporary files. Default: *xplore\_home/dsearch/cps/cps\_daemon/temp*.
- *temp\_file\_folder*: Directory for temporary format and language identification. Default: *xplore\_home/dsearch/cps/cps\_daemon/temp*.
- *keep\_temp\_file*: Whether to keep CPS temporary files (true) or not (false).
- *temp\_file\_retain\_time*: How long will CPS temporary files be retained. This parameter is only effective when *keep\_temp\_file* is set to false.
- *daemon\_restart\_memory\_threshold*: Maximum memory consumption at which CPS is restarted.
- *use\_direct\_io*: Requires CPS to read and write to devices directly.
- *io\_block\_unit*: Logical block unit of the read/write target device.
- *io\_chunk\_size*: Size for each read/write chunk.
- *cut\_off\_text*: Set to true to cut off text of large documents that exceed *max\_text\_threshold* instead of rejecting entire document. Changes to this setting require an index rebuild.
- *Language*: Languages supported by CPS, delimited by comma. Changes to this setting require an index rebuild.

## A.4 Search service configuration parameters

You can configure the following settings for the search service in xPlore administrator. The default values have been optimized for most environments.

- query-default-locale: Default locale for queries. See Basistech documentation for identified language codes. See your release notes for supported languages in this release. Default: en (English).
- query-default-result-batch-size: Default size of result batches that are sent to the client. Default: 200. In a Documentum environment, dfc.batch\_hint\_size in dfc.properties overrides this setting. Negative values default to a single batch.
- query-result-cache-size: Default size of results buffer. When this limit is reached, no more results are fetched from xDB until the client asks for more results. Default: 200.
- query-result-spool-location: Path to location at which to spool results. Default: *xplore\_home/dsearch/spool*
- query-default-timeout: Interval in milliseconds for a query to time out. Default: 60000 (one minute). Negative values default to no timeout (not recommended). This setting is overridden by the query\_timeout parameter in dm\_ftengine\_config and by the Search Service and IDFXQuery TIMEOUT APIs.
- query-threadpool-size: Number of threads for the query thread pool. Default: 2 × number of OS processors.
- query-thread-sync-interval: Used for xPlore internal synchronization. Interval after which results fetching is suspended when the result cache is full. For a value of 0, the thread waits indefinitely until space is available in the cache (freed up when the client application retrieves results). Default: 100 units.
- query-thread-max-idle-interval: Query thread is freed up for reuse after this interval, because the client application has not retrieved the result. (Threads are freed immediately after a result is retrieved.) Default: 3600000.
- query-summary-default-highlighter: Class that determines summary and highlighting. Default: com.emc.documentum.core.fulltext.indexserver.services.summary.DefaultSummary. For query summary configuration, see “[Configuring query summaries](#)” on page 259.
- query-summary-display-length: Number of characters to return as a dynamic summary. Default: 256.
- query-summary-highlight-begin-tag: HTML tag to insert at beginning of summary. Default: empty string.
- query-summary-highlight-end-tag: HTML tag to insert at end of summary. Default: empty string.
- query-enable-dynamic-summary: If context is not important, set to false to return as a summary the first *n* chars defined by the query-summary-display-length

configuration parameter. For summaries evaluated in context, set to true (default).

- query-index-covering-values: Supports Documentum DQL evaluation. Do not change unless tech support directs you to.
- query-facet-max-result-size: Documentum only. Sets the maximum number of results used to compute facet values. For example, if query-facet-max-result-size=12, only 12 results for all facets in a query are returned. If a query has many facets, the number of results per facet is reduced accordingly. Default: 10000.
- query-discard-unselective-wildcard-query: Discards unselective XQuery expressions by default. To enable unselective wildcard xqueries, set to false and add `xdb.lucene.noTokenFtcontainsToMatchAll=true` to `xdb.properties`. The unselective wildcard query pattern is as follows:

```
ftcontains ".*"
ftcontains "@.*"
ftcontains " "
ftcontains ".."
```

- query-parallel-execution: Set this value to true to run all queries against multiple collections in parallel. When this is set to true, a log message “Use Parallel Execution” will be recorded for each query logged in the `dsearch.log` file at the INFO level.

When there are too many queries running in parallel concurrently, a query may be forced to run in nonparallel mode to prevent system overload. In this case, the following message is logged for the query at the INFO level: “Current active parallel thread count too high to allow parallel query execution. Running in nonparallel for the query”.

When the parallel search resource pool is exhausted, the system will attempt to retry the query request for up to the number of times specified by the `query-executor-retry-limit` property value in `search-config`. The amount of wait time between retry attempts is determined by the `query-executor-retry-interval` property setting.

- query-summary-fragment-size: Number of characters to return as a fragment. The maximum number of fragments a query can return equals `query-summary-display-length/query-summary-fragment-size`. Default is 64.
- query-parallel-summary-calculation: Set to true to enable parallel summary calculation. Default: false.
- query-parallel-execution-thread-pool-size: The maximum thread pool size for parallel execution. Default: 100.
- query-parallel-summary-calculation-thread-pool-size: The maximum thread pool size for parallel summary calculation. Default: 100.
- query-summary-process-all-highlight-nodes: Set it to true to enable summary highlighting for multiple elements. By default, this parameter is not provided in `indexserverconfig.xml`.

- query-highlight-based-on-phrase: Set to true to enable highlighting based on the exact phrase, or sequence of terms, for a phrase query. Set to false to enable highlighting based on each term for a phrase query or term query.

## A.5 API Reference

### CPS APIs

Content processing service APIs are available in the interface IFtAdminCPS in the package com.emc.documentum.core.fulltext.client.admin.api.interfaces. This package is in the SDK jar file dsearchadmin-api.jar.

To add a CPS instance using the API addCPS(String instanceName, URL url, String usage), the following values are valid for usage: all, index, or search. If the instance is used for CPS alone, use *index*. For example:

```
addCPS("primary",
http://myhost:9700/cps/ContentProcessingService?wsdl",
index")
```

*CPS configuration keys for setCPSCfg()*

- CPS-requests-max-size: Maximum size of CPS queue
- CPS-requests-batch-size: Maximum number of CPS requests in a batch
- CPS-thread-wait-time: Maximum wait time in milliseconds to accumulate requests in a batch
- CPS-executor-queue-size: Maximum size of CPS executor queue before spawning a new worker thread

### Indexing engine APIs (DFC)

DFC exposes APIs to get information about the index engine that a repository uses. Get the IDfFtConfig interface from a DFC session. Use IDfFtConfig to get the following information:

- getCollectionPath. Returns the complete path of the root collection, useful for constructing some queries.
- getEngine(). Returns DSEARCH, FAST, Lucene, or unknown
- getEngineConfig(). The following example assumes that a connection to the repository has been established and saved in the class variable *m\_session* (instance of IDFSession).

```
import com.documentum.fc.client.fulltext.IDfFtConfig;
public void getEngineConfig() throws Exception
{
    IDfFtConfig cfg = m_session.getFtConfig();
    String coll = cfg.getCollectionPath();
    System.out.println("Collection path: " + coll);
    String eng = cfg.getEngine();
    System.out.println("Full-text engine: " + eng);
}
```

- `isCapabilitySupported(String capability)`. Supported inputs are `scope_search`, `security_eval_in_fulltext`, `xquery`, `relevance_ranking`, `hit_count`, and `search_topic`. Returns 1 for supported, 0 for unsupported, and -1 when it cannot be determined. Supported returns are as follows:
  - `scope_search`: XML element data can be searched, like the <IN> operator in previous versions of Documentum Server. xPlore and FAST support this feature.
  - `security_eval_in_fulltext`: Security is evaluated in the full-text engine before results are returned to Documentum Server, resulting in faster query results. This feature is available only in xPlore.
  - `xquery`: xPlore supports XQuery syntax.
  - `relevance_ranking`: The full-text engine scores results using configurable criteria. xPlore and FAST support this feature.
  - `hit_count`: The full-text engine returns the total number of hits before returning results. FAST supports this feature; xPlore does not support it.



- Note:** The count that is returned for DQL queries does not reflect the application of security, which reduces the actual count returned for the query.
- `search_topic`: The full-text engine indexes all XML elements and their attributes. FAST supports zone searching (search topic) for backward compatibility. The xPlore server returns false.

For information on creating and retrieving facets, see the Facets chapter.

## Search APIs

Search service APIs are available in the following packages of the SDK jar file `dsearchadmin-api.jar` :

- `IFtAdminSearch` in the package `com.emc.documentum.core.fulltext.client.admin.api.interface`.
- `IFtSearchSession` in `com.emc.documentum.core.fulltext.client.search`
- `IFtQueryOptions` in `com.emc.documentum.core.fulltext.common.search`.

## Auditing APIs

Auditing APIs are available in the interface `IFtAdminAudit` in the package `com.emc.documentum.core.fulltext.client.admin.api.interfaces`. This package is in the SDK jar file `dsearchadmin-api.jar`.

## **xDB data management APIs**

The data management APIs are available in the interface IFtAdminDataManagement in the package com.emc.documentum.core.fulltext.client.admin.api.interfaces. This package is in the SDK jar file dsearchadmin-api.jar.



# Appendix B. The dftxml Category

## B.1 Extensible Documentum DTD

### Viewing the dftxml representation of a document

Documentum repository content is stored in XML format. Customer-defined elements and attributes can be added to the DTD as children of *dmftcustom*. Each element specifies an attribute of the object type. The object type is the element in the path dmftdoc/dmftmetadata/*type\_name*, for example, dmftdoc/dmftmetadata/dm\_document.

To view the dftxml representation that is generated by the index agent, add the following element as a child of the exporter element in `indexagent.xml` in `xplore_home/<wildfly_version>/server/DctmServer_Indexagent/deployments/IndexAgent.war/WEB-INF/classes`.

```
<keep_dftxml>true</keep_dftxml>
```

To view the dftxml representation of a document that has been indexed, open xPlore administrator and click the document in the collection view.

To find the path of a specific attribute in dftxml, use a Documentum client to look up the object ID of a custom object. Using xPlore administrator, open the target collection and paste the object ID into the **Filter word** box. Click the resulting document to see the dftxml representation.

### DTD

This DTD is subject to change. Following are the top-level elements under *dmftdoc*.

**Table B-1: dftxml top-level elements**

Element	Description
dmftkey	Contains Documentum object ID (r_object_id)
dmftmetadata	Contains elements for all indexable attributes from the standard Documentum object model, including custom object types. Each attribute is modeled as an element and value. Repeating attributes repeat the element name and contain a unique value. Some metadata, such as r_object_id, are repeated in other elements as noted.
dmftvstamp	Contains the internal version stamp (i_vstamp) attribute.

Element	Description
dmftsecurity	Contains security attributes from the object model plus computed attributes: acl_name, acl_domain, and ispublic.
dmftinternal	Contains attributes used internally for query processing.
dmftversions	Contains version labels and <i>iscurrent</i> for sysobjects.
dmftfolders	Contains the folder ID and folder parents.
dmftcontents	Contains content-related attributes and one or more pointers to content files. The actual content can be stored within the child element dmftcontent as a CDATA section.
dmftcustom	Contains searchable information supplied by custom applications. Requires a TBO. See “Injecting data and supporting joins” on page 108.
dmftsearchinternals	Contains tokens used by static and dynamic summaries.

 **Example B-1: Example dftxml of a custom object type**

```
<?xml version="1.0"?>
<dmftdoc dmftkey="090a0d6880008848" dss_tokens=":dftxml:1">
  <dmftkey>090a0d6880008848</dmftkey>
  <dmftmetadata>
    <dm_sysobject>
      <r_object_id dmfttype="dmid">090a0d6880008848</r_object_id>
      <object_name dmfttype="dmstring">mylog.txt</object_name>
      <r_object_type dmfttype="dmstring">techpubs</r_object_type>
      <r_creation_date dmfttype="dmdate">2010-04-09T21:40:47</r_creation_date>
      <r_modify_date dmfttype="dmdate">2010-04-09T21:40:47</r_modify_date>
      <r_modifier dmfttype="dmstring">Administrator</r_modifier>
      <r_access_date dmfttype="dmdate"/>
      <a_is_hidden dmfttype="dmbool">false</a_is_hidden>
      <i_is_deleted dmfttype="dmbool">false</i_is_deleted>
      <a_retention_date dmfttype="dmdate"/>
      <a_archive dmfttype="dmbool">false</a_archive>
      <a_link_resolved dmfttype="dmbool">false</a_link_resolved>
      <i_reference_cnt dmfttype="dmint">1</i_reference_cnt>
      <i_has_folder dmfttype="dmbool">true</i_has_folder>
      <i_folder_id dmfttype="dmid">0c0a0d6880000105</i_folder_id>
      <r_link_cnt dmfttype="dmint">0</r_link_cnt>
      <r_link_high_cnt dmfttype="dmint">0</r_link_high_cnt>
      <r_assembled_from_id dmfttype="dmid">0000000000000000</r_assembled_from_id>
      <r_frzn_assembly_cnt dmfttype="dmint">0</r_frzn_assembly_cnt>
      <r_has_frzn_assembly dmfttype="dmbool">false</r_has_frzn_assembly>
      <r_is_virtual_doc dmfttype="dmint">0</r_is_virtual_doc>
      <i_contents_id dmfttype="dmid">060a0d688000ec61</i_contents_id>
      <a_content_type dmfttype="dmstring">crttext</a_content_type>
      <r_page_cnt dmfttype="dmint">1</r_page_cnt>
      <r_content_size dmfttype="dmint">130524</r_content_size>
      <a_full_text dmfttype="dmbool">true</a_full_text>
      <a_storage_type dmfttype="dmstring">filestore_01</a_storage_type>
      <i_cabinet_id dmfttype="dmid">0c0a0d6880000105</i_cabinet_id>
      <owner_name dmfttype="dmstring">Administrator</owner_name>
      <owner_permit dmfttype="dmint">7</owner_permit>
    </dm_sysobject>
  </dmftmetadata>
</dmftdoc>
```

```

<group_name dmfttype="dmstring">docu</group_name>
<group_permit dmfttype="dmint">5</group_permit>
<world_permit dmfttype="dmint">3</world_permit>
<i_antecedent_id dmfttype="dmid">0000000000000000</i_antecedent_id>
<i_chronicle_id dmfttype="dmid">090a0d6880008848</i_chronicle_id>
<i_latest_flag dmfttype="dmbool">true</i_latest_flag>
<r_lock_date dmfttype="dmdate"/>
<r_version_label dmfttype="dmstring">1.0</r_version_label>
<r_version_label dmfttype="dmstring">CURRENT</r_version_label>
<i_branch_cnt dmfttype="dmint">0</i_branch_cnt>
<i_direct_dsc dmfttype="dmbool">false</i_direct_dsc>
<r_immutable_flag dmfttype="dmbool">false</r_immutable_flag>
<r_frozen_flag dmfttype="dmbool">false</r_frozen_flag>
<r_has_events dmfttype="dmbool">false</r_has_events>
<acl_domain dmfttype="dmstring">Administrator</acl_domain>
<acl_name dmfttype="dmstring">dm_450a0d6880000101</acl_name>
<i_is_reference dmfttype="dmbool">false</i_is_reference>
<r_creator_name dmfttype="dmstring">Administrator</r_creator_name>
<r_is_public dmfttype="dmbool">true</r_is_public>
<r_policy_id dmfttype="dmid">0000000000000000</r_policy_id>
<r_resume_state dmfttype="dmint">0</r_resume_state>
<r_current_state dmfttype="dmint">0</r_current_state>
<r_alias_set_id dmfttype="dmid">0000000000000000</r_alias_set_id>
<a_is_template dmfttype="dmbool">false</a_is_template>
<r_full_content_size dmfttype="dmdouble">130524</r_full_content_size>
<a_is_signed dmfttype="dmbool">false</a_is_signed>
<a_last_review_date dmfttype="dmdate"/>
<i_retain_until dmfttype="dmdate"/>
<i_partition dmfttype="dmint">0</i_partition>
<i_is_replica dmfttype="dmbool">false</i_is_replica>
<i_vstamp dmfttype="dmint">0</i_vstamp>
<webpublish dmfttype="dmbool">false</webpublish>
</dm_sysobject>
</dmftmetadata>
<dmftvstamp>
  <i_vstamp dmfttype="dmint">0</i_vstamp>
</dmftvstamp>
<dmftsecurity>
  <acl_name dmfttype="dmstring">dm_450a0d6880000101</acl_name>
  <acl_domain dmfttype="dmstring">Administrator</acl_domain>
  <isppublic dmfttype="dmbool">true</isppublic>
</dmftsecurity>
<dmftinternal>
  <docbase_id dmfttype="dmstring">658792</docbase_id>
  <server_config_name dmfttype="dmstring">DSS_LH1</server_config_name>
  <contentid dmfttype="dmid">060a0d688000ec61</contentid>
  <r_object_id dmfttype="dmid">090a0d6880008848</r_object_id>
  <r_object_type dmfttype="dmstring">techpubs</r_object_type>
  <i_all_types dmfttype="dmid">030a0d68800001d7</i_all_types>
  <i_all_types dmfttype="dmid">030a0d6880000129</i_all_types>
  <i_all_types dmfttype="dmid">030a0d6880000105</i_all_types>
  <i_dftxml_schema_version dmfttype="dmstring">5.3</i_dftxml_schema_version>
</dmftinternal>
<dmftversions>
  <r_version_label dmfttype="dmstring">1.0</r_version_label>
  <r_version_label dmfttype="dmstring">CURRENT</r_version_label>
  <iscurrent dmfttype="dmbool">true</iscurrent>
</dmftversions>
<dmftfolders>
  <i_folder_id dmfttype="dmid">0c0a0d6880000105</i_folder_id>
</dmftfolders>
<dmftcontents>
  <dmftcontent>
    <dmftcontentattrs>
      <r_object_id dmfttype="dmid">060a0d688000ec61</r_object_id>
      <page dmfttype="dmint">0</page>
      <i_full_format dmfttype="dmstring">crtext</i_full_format>
    </dmftcontentattrs>
    <dmftcontentref content-type="text/plain" islocalcopy="true" lang="en" encoding="US-ASCII" summary_tokens="dmftsummarytokens_0">

```

```

<![CDATA[...]]>
</dmftcontentref>
</dmftcontent>
</dmftcontents>
<dmftsearchinternals dss_tokens="excluded">
<dmftstaticsummarytext dss_tokens="excluded"><![CDATA[mylog.txt ]]>
</dmftstaticsummarytext>
<dmftsummarytokens_0 dss_tokens="excluded"><![CDATA[1Tkns ...]]>
</dmftsummarytokens_0></dmftsearchinternals></dmftdoc>

```

The attribute *islocalcopy* indicates whether the content was indexed. If *true*, only the metadata was indexed, and no copy of the content exists in the index.



## B.2 Supporting XML namespaces

Although dftxml documents created from Documentum objects do not contain namespaces, the dftxml category does support XML namespaces so that the xPlore indexing and query services can be integrated with other content management systems other than Documentum Server that require separation and identification of duplicate XML element names within an XML document.

xPlore indexing and query services fully supports XML namespaces from end to end – Indexing, storing, and querying XML documents with namespaces, configuring index paths that contain namespaces, and executing XQueries with namespace mappings.

Include XML namespaces at the following configuration points to take advantage of this feature:

- Text extraction configuration

When specifying an XPath value to the element whose content requires text extraction for indexing as the value of the path attribute, include the namespace in the *path* attribute on the *for-element-with-name* element; for example:

```
<do-text-extraction>... <for-element-with-name path="/dmftdoc/
dmftcontents/dmftcontent/ {http://www.te.com}dmftcontentref">...</do-
text-extraction>
```

- Subpath definition

Include namespaces in subpath definitions to specify paths to elements with namespaces. For example, to specify a path to the following element with a namespace:

```
<com:product xmlns:com="http://www.opentext.com"
dmfttype="dmstring">xPlore</com:product>
```

Define the subpath as follows:

```
<sub-path description="leading wildcard queries" returning-
contents="false" value-comparison="true" full-text-search="true"
enumerate-repeating-elements="false" leading-wildcard="true"
type="string" path="dmftmetadata//{http://www.opentext.com}product"/>
```

Include the namespace in your XQuery expression to perform the search:

```
declare namespace com='http://www.opentext.com'; for $doc in  
/dmftdoc[dmftmetadata//com:product ftcontains 'xPlore' with  
stemming]...
```

The search will only return matching results with the specified namespace defined in the subpath definitions.

- Linguistic processing related configurations

When performing linguistic processing related configurations such as static summaries and language identification, you can include elements with namespaces for CPS to handle such documents.

For example, for the following element with a namespace in the dftxml document:

```
<ngis:subject xmlns:ngis="http://www.ngis.com"  
dmfttype="dmstring">subject with namespace</ngis:subject>
```

To define the *subject* element as a static summary element, in indexserverconfig.xml, configure the setting as follows:

```
... <elements-for-static-summary max-size="65536"> <element-name  
name="{http://www.ngis.com}subject"/></elements-for-static-summary>...
```

To configure the *subject* element to be used for language identification, in indexserverconfig.xml, configure the setting as follows:

```
...<linguistic-process> <element-for-language-idenfication  
name="{http://www.ngis.com}subject"/></linguistic-process>...
```

After the configuration, CPS will only process *subject* elements with the *ngis* namespace but not *subject* elements with no or different namespaces.



# Appendix C. XQuery and VQL Reference

## C.1 Tracking XQueries

You can issue the following XQuery expressions against the tracking database for each domain. Many of these expressions are available in xPlore administrator or as audit reports. These XQuery expressions can be submitted in the xDB console.

### Object count from tracking DB

*Get object count in a collection*

```
count(//trackinginfo/document[collection-name=""])
```

For example:

```
for $i in collection("dsearch/SystemInfo")
return count($i//trackinginfo/document)
```

*Get object count in all collections (all indexed objects)*

```
count(//trackinginfo/document)
```

For example:

*Get object count in library*

```
count(//trackinginfo/document[library-path=""])
```

### Find documents

*Find collection in which a document is indexed*

```
//trackinginfo/document[@id=""]/collection-name/string(.)
```

For example:

```
for $i in collection("dsearch/SystemInfo")
where $i//trackinginfo/document[@id="TestCustomType_txt1276106246060"]
return $i//trackinginfo/document/collection-name
```

*Find library in which a document is indexed*

```
//trackinginfo/document[@id=""]/library-path/string(.)
```

Get tracking information for a document

```
//trackinginfo/document[@id=""]
```

## C.2 VQL and XQuery Syntax Equivalents

xPlore does not support the Verity Query Language (VQL). The following table maps VQL syntax examples that have equivalent in XQuery.

**Table C-1: DQL and XQuery mapping**

DQL	XQuery
IN	<pre>for \$i in collection(' /XX/dsearch/Data')/dmftdoc[ (dmftcontents/dmftcontent ftcontains ('test1')) ]</pre>
NEAR/N	<pre>for \$i in collection(' /XX/dsearch/Data')/dmftdoc[ (dmftcontents/dmftcontent ftcontains ('test1' ftand 'test2' distance exactly N words)) ]</pre>
ORDERED	<pre>for \$i in collection(' /XX/dsearch/Data')/dmftdoc[ (dmftcontents/dmftcontent ftcontains ('test1' ftand 'test2') ordered)]</pre>
ENDS	<pre>let \$result := ( for \$i in collection(' /XX/dsearch/Data')/dmftdoc[ (dmftcontents/dmftcontent ftcontains ('test1')) and (ends-with(dmftmetadata/dm_sysobject/ object_name, 'test2'))])</pre>
STARTS	<pre>for \$i in collection(' /XX/dsearch/Data')/dmftdoc[ (dmftcontents/dmftcontent ftcontains ('test1')) and starts-with(dmftinternal/r_object_type, 'dm_docu')]</pre>

## Appendix D. xPlore Glossary

Term	Description
category	A category defines a class of documents and their XML structure.
collection	A collection is a logical group of XML documents that is physically stored in an xDB library. A collection represents the most granular data management unit within xPlore.
content processing service (CPS)	The content processing service (CPS) retrieves indexable content from content sources and determines the document format and primary language. CPS parses the content into index tokens that xPlore can process into full-text indexes.
domain	A domain is a separate, independent group of collections with an xPlore deployment.
DQL	Documentum Query Language, used by many Documentum Server clients
FTDQL	Full-text Documentum Query Language
<i>ftintegrity</i>	A standalone Java program that checks index integrity against Documentum Server repository documents. The <i>ftintegrity</i> script calls the <i>state of index</i> job in the Documentum Server.
full-text index	Index structure that tracks terms and their occurrence in a document.
index agent	Documentum application that receives indexing requests from the Documentum Server. The agent prepares and submits an XML representation of the document to xPlore for indexing.
ingestion	Process in which xPlore receives an XML representation of a document and processes it into an index.
instance	A xPlore instance is one deployment of the xPlore WAR file to an application server container. You can have multiple instances on the same host (vertical scaling), although it is more common to have one xPlore instance per host (horizontal scaling). The following processes can run in an xPlore instance: CPS, indexing, search, xPlore administrator. xPlore can have multiple instances installed on the same host.

Term	Description
lemmatization	Lemmatization is a normalization process in which the lemmatizer finds a canonical or dictionary form for a word, called a <i>lemma</i> . Content that is indexed is also lemmatized unless lemmatization is turned off. Terms in search queries are also lemmatized unless lemmatization is turned off.
Lucene	Apache open-source, Java-based full-text indexing, and search engine.
node	In xPlore and xDB, <i>node</i> is sometimes used to denote <i>instance</i> . It does not denote <i>host</i> .
persistence library	Saves CPS, indexing, and search metrics. Configurable in indexserverconfig.xml.
<i>state of index</i> job	Documentum Server configuration installs the state of index job dm_FTStateOfIndex. This job is run from Documentum Administrator. The <i>fintegrity</i> script calls this job, which reports on index completeness, status, and indexing failures.
status library	A status library reports on indexing status for a domain. There is one status library for each domain.
stop words	Stop words are common words filtered out of queries to improve query performance. Stop words can be searched when used in a phrase.
text extraction	Identification of terms in a content file.
token	Piece of an input string defined by semantic processing rules.
tracking library	An xDB library that records the object IDs and location of content that has been indexed. There is one tracking database for each domain.
transactional support	Small in-memory indexes are created in rapid transactional updates, then merged into larger indexes. When an index is written to disk, it is considered clean. Committed and uncommitted data before the merge is searchable along with the on-disk index.

Term	Description
watchdog service	Installed by the xPlore installer, the watchdog service pings all xPlore instances and sends an email notification to the administrator when an instance does not respond. The watchdog service can also be configured to automatically restart the index agent when it has stopped working.
xDB	xDB is a database that enables high-speed storage and manipulation of many XML documents. In xPlore, an xDB library stores a collection as a Lucene index and manages the indexes on the collection. The XML content of indexed documents can optionally be stored.
XQFT	W3C full-text XQuery and XPath extensions described in the XQuery and XPath Full Text 1.0 specification. Support for XQFT includes logical full-text operators, wildcard option, anyall option, positional filters, and score variables.
XQuery	W3C standard query language that is designed to query XML data. xPlore receives xQuery expressions that are compliant with the XQuery standard and returns results.

