



Verity Command-Line Indexing Reference

Version 6.0

August 19, 2005
Part Number DM0680

Verity, Incorporated
894 Ross Drive
Sunnyvale, California 94089
(408) 541-1500

Verity Benelux BV
Coltbaan 31
3439 NG Nieuwegein
The Netherlands

Copyright 2005 Verity, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of the copyright owner, Verity, Inc., 894 Ross Drive, Sunnyvale, California 94089. The copyrighted software that accompanies this manual is licensed to the End User for use only in strict accordance with the End User License Agreement, which the Licensee should read carefully before commencing use of the software.

Verity®, Ultraseek®, TOPIC®, KeyView®, and Knowledge Organizer® are registered trademarks of Verity, Inc. in the United States and other countries. The Verity logo, Verity Portal One™, and Verity® Profiler™ are trademarks of Verity, Inc.

Portions of this product Copyright 2003, Sun Microsystems, Inc. All rights reserved. Use is subject to license terms. Sun, Sun Microsystems, the Sun logo, Solaris, Java, the Java Coffee Cup logo, J2SE, and all trademarks and logos based on Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Xerces XML Parser Copyright 1999-2000 The Apache Software Foundation. All rights reserved.

Microsoft is a registered trademark, and MS-DOS, Windows, Windows 95, Windows NT, and other Microsoft products referenced herein are trademarks of Microsoft Corporation.

IBM is a registered trademark of International Business Machines Corporation.

WordNet 1.7 Copyright © 2001 by Princeton University. All rights reserved

Includes Adobe® PDF. Adobe is a trademark of Adobe Systems Incorporated.

Portions of this product use Teragram Software.

Includes IBM's XML Parser for C++ Edition.

Includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product may incorporate intellectual property owned by Microsoft Corporation. The terms and conditions upon which Microsoft is licensing such intellectual property may be found at

<http://msdn.microsoft.com/library/en-us/odcXMLRef/html/odcXMLRefLegalNotice.asp?frame=true>

All other trademarks are the property of their respective owners.

Notice to Government End Users

If this product is acquired under the terms of a **DoD contract**: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of 252.227-7013. **Civilian agency contract**: Use, reproduction or disclosure is subject to 52.227-19 (a) through (d) and restrictions set forth in the accompanying end user agreement. Unpublished-rights reserved under the copyright laws of the United States. Verity, Inc., 894 Ross Drive Sunnyvale, California 94089.

Contents

Figures, Tables, and Listings..... 15

Preface 17

Using This Book 17

 Version 17

 Organization of This Book 18

 Stylistic Conventions..... 19

Related Documentation 21

Verity Technical Support..... 21

PART I OVERVIEW

1 Verity Indexers Overview..... 25

 Verity Spider..... 25

 Verity K2 Spider..... 26

 The mkvdk Command-Line Tool..... 26

PART II VERITY SPIDER

2 Verity Spider Overview 29

 Verity Spider Features..... 30

 State Maintenance Through a Persistent Store..... 30

 Performance..... 30

 Automatic Collection Optimization 31

 How Verity Spider Indexes 32

3 Verity Spider Reference	35
Verity Spider Syntax	36
Overview	36
The Verity Spider Command	36
Using a Command File	37
Indexing Job Scope	37
Reference of Command-Line Options	38
Initialization Options	41
-start	41
Gateway Type is Web (HTTP Gateway)	41
Gateway Type is File (File System Gateway)	42
Gateway Type is Notes (Lotus Notes Gateway)	42
Gateway Type is ODBC (ODBC Gateway)	42
Gateway Type is Documentum (Documentum Gateway)	42
Gateway Type is Exchange (Exchange Gateway)	43
-refresh	44
-refreshtime	45
-reparse	45
-restart	46
Core Options	47
-cmdfile	47
-collection	48
-help	48
-jobpath	48
-style	49
Gateway Options	50
-notes -odbc -documentum -exchange	50
-gateway	50
Processing Options	51
-abspath	51
-checkacl	51
-detectdupfile	52
-indexers	52
-license	52
-maxfilehandles	53

-maxindmem	53
-maxnumdoc	53
-mimemap.....	54
-nodupdetect	54
-noindex	55
-nosubmit.....	55
-numpages	56
-persist.....	56
-preferred	57
-prefixmap	58
-processbif.....	58
-regexp.....	59
-submitsize.....	59
-temp.....	60
Networking Options	61
-agentname	61
-connections.....	61
-delay	62
-header.....	62
-hostcache	62
-noflowctrl	63
-noproxy.....	64
-proxy	65
-proxyauth	65
-retry	66
-timeout	66
Paths and URLs Options	67
-anonymous.....	67
-auth.....	67
-auth2.....	68
-certfile.....	68
-cgiok	69
-domain	69
-followdup	70
-followsymlink	70

Contents

-host	70
-https	71
-jumps	71
-keyfile	71
-nodocrobo	72
-nofollow	72
-norobo	73
-pathlen	73
-unlimited	74
-useget	74
-virtualhost	74
Content Options	75
-casesen	75
-exclude	75
-include	76
-indexclude	77
-indinclude	78
-indmimeexclude	79
-indmimeinclude	80
-indskip	81
-maxdocsize	82
-metafile	82
-mimeexclude	83
-mimeinclude	84
-mindocsize	84
-query	85
-skip	86
Locale Options	87
-charmap	87
-common	87
-datefmt	87
-keyencode	88
-language	88
-locale	88
-msgdb	89

-repos_charmap	89
Logging Options	90
-loglevel.....	90
-debug -trace -verbose.....	91
Maintenance Options	92
-nooptimize.....	92
-purge	92
-repair	93
Using vsdb	94
vsdb Arguments	94
vsdb Examples	98
Purging Duplicate Documents	98
Removing Duplicate Documents from Searches	101
Restoring a Corrupted Persistent Store.....	103
Evaluating “include” and “exclude” Criteria.....	104
Evaluation Workflow	104
Example Indexing Job	105
Verity Spider Command	105
Candidates from Starting Point.....	105
Evaluating the Candidates	106
The Use of Last-Modified Date	111
How Last-Modified is Used	111
4 Verity Spider Examples	113
Examples	114
Deleting Linked Documents	115
Skipping Documents.....	116
Preferring a Site for Duplicates.....	117
Reparsing a Site.....	118
Omitting Optimization	119
Indexing Virtual Hosts.....	120
Updating Only Certain Documents.....	121
Custom Value for Last-Modified Date	122
An Intranet with CGI.....	123
Web Sites and Proxy Servers	124

Adding to an Existing Collection	125
Including Previously Dropped Documents.....	126
File Systems	127
Specific Situations and Concepts.....	129
Customizing the Last-Modified Date	130
Indexing Secure Web Sites	134
Procedure.....	134
The vgwhttp.cfg File	136
Sample vgwhttp.cfg File.....	139
Indexing with Proxy Servers.....	140
The vgwhttp.cfg File	141
Sample vgwhttp.cfg File.....	143
Indexing Network, UNC and Secure Paths on Windows	144
Running vspider.exe	144
Defining the Authenticating Repository for Secure File System Collections	145
Access Control List (ACL) Caching — Domain versus Local Users.....	146
Prefix Mapping	147
Setting MIME Types.....	152
Default MIME Types.....	152
MIME Types Mapped by File Extension.....	153
Complete List of Known MIME Types	154
Indexing Unknown MIME Types	155
MIME Types and Web Crawling	156
MIME Types and File System Indexing.....	156
Syntax Restrictions	157
Using StyleSet Editor in Standalone Mode.....	158

PART III K2 SPIDER

5 K2 Spider Overview	161
K2 Spider Features.....	162
Overview of Features.....	162
K2 Spider System.....	165
Component Limitations.....	165
K2 Spider System Illustration	166

6	Building a K2 Spider System.....	167
	K2 Spider Configurations.....	168
	Binding Crawlers with Indexers.....	169
	Binding Tips	169
	Multiple Processes on a Single Host	170
	Separate Controller with One Other Host	171
	Fully Distributing All Components	173
	Configuration Considerations	175
7	Running K2 Spider Server	177
	Running a Controller.....	178
	Using k2spider_srv as a Controller.....	178
	Options for Using k2spider_srv as a Controller	178
	Running a Crawler	184
	Using k2spider_srv as a Crawler	184
	Options for Using k2spider_srv as a Crawler	185
	Running an Indexer.....	193
	Using k2spider_srv as an Indexer	193
	Options for Using k2spider_srv as an Indexer.....	193
	-numpages	201
	k2spider_srv Reference	206
8	Using K2 Spider Client	209
	Starting the K2 Spider Client.....	210
	Migrating K2 Spider Jobs	210
	Options for Using k2spider_srv as a Controller	211
	Using k2spider_cli to Index.....	222
	Creating Indexing Jobs	222
	Managing Indexing Jobs.....	223
	Managing k2spider_srv Services.....	226
	Shutting Down Services.....	226
	Using the Bulk Submit Feature.....	227
	Connecting to a Controller	227
	Defining Document-Related Fields.....	227

Creating a Bulk Insert File (BIF)	228
Character Set of Bulk Insert File (BIF)	230
Submitting the Bulk Insert File	233
Submit Actions.....	233
Deleting Keys and Records with VSDB.....	234
Reporting.....	236
Connecting to a Controller	236
Using getinfo	236
Using getstat.....	238
Using VSDB	240
Sample VSDB Session	241
VSDB Summaries	242
VSDB Actions	242
VSDB Options	243
k2spider_cli Reference	246
9 Job Initialization File Reference	249
General Conventions.....	250
Summary of INI File Options.....	252
Job Section.....	254
FormsAuth Section	268
Crawlers Section	270
Indexers Section	271
Network Section.....	272
Schedule Section	274
Proxy Section	277
DocRouter Section	279
Criterion Section	287
Sample myjob.ini File.....	307
Job Section.....	308
Forms Based Authentication Section	309
Crawlers, Indexers, Network and Schedule Sections.....	310
Proxy and DocRouter Sections	311
Crawling and Indexing Criteria Section — Examples	312
Criterion Section	313

Evaluating “include” and “exclude” Criteria.....	314
Evaluation Workflow	314
Sample Indexing Job	315
The Use of Last-Modified Date	321
10 K2 Spider Examples	323
Examples	323
Indexing Workflow	324
Indexing a Notes Repository.....	325
Indexing an ODBC Repository	329
Indexing a Documentum Repository	332
Specific Situations and Concepts.....	335
Customizing the Last-Modified Date	336
Scenarios for Customizing	336
Indexing Secure Web Sites	339
Procedure.....	339
The vgwhttp.cfg File	340
Sample vgwhttp.cfg File.....	343
Indexing with Proxy Servers.....	344
Procedure.....	344
The vgwhttp.cfg File	345
Sample vgwhttp.cfg File.....	347
Indexing Network, UNC and Secure Paths on Windows	348
Running k2spider_srv.exe.....	348
Indexing Network, UNC and Secure Paths on UNIX	349
Defining the Authenticating Repository for Secure File System Collections	350
Access Control List (ACL) Caching — Domain versus Local Users	351
Prefix Mapping	352
K2 Spider Indexing Options	352
Using the Control File.....	352
Setting MIME Types.....	356
Default MIME Types.....	356
MIME Types Mapped by File Extension.....	356
Complete List of Known MIME Types	357
Indexing Unknown MIME Types	358
MIME Types and Web Crawling	359
MIME Types and File System Indexing	360

Contents

Syntax Restrictions	360
Using StyleSet Editor in Standalone Mode	361
Index	371

Figures, Tables, and Listings

Figure 2-1	Verity Spider indexing in action.....	32
Figure 3-1	Workflow of Verity Spider evaluating criteria.....	104
Table 4-1	Case-specific Options	116
Table 4-2	Case-specific Options	117
Table 4-3	Case-specific Options	118
Table 4-4	Unnecessary Options for this Case.....	118
Table 4-5	Case-specific Options	119
Table 4-6	Case-specific Options	120
Table 4-7	Case-specific Options	121
Table 4-8	Case-specific Options	122
Table 4-9	Case-specific Options	123
Table 4-10	Unnecessary Options for this Case.....	123
Table 4-11	Case-specific Options	124
Table 4-12	Case-specific Options	125
Table 4-13	Unnecessary Options for this Case.....	125
Table 4-14	Case-specific Options	126
Table 4-15	Case-specific Options	128
Table 4-16	Unnecessary Options for this Case.....	128
Table 4-17	Case-specific Options	131
Figure 5-1	A distributed K2 Spider indexing system	166
Figure 6-1	Multiple processes on a single host.....	170

Figure 6-2	Separate Controller with one other host.....	171
Figure 6-3	Extending a K2 Spider indexing system	172
Figure 6-4	Fully distributing all components.....	173
Figure 8-1	K2 Spider Client session	211
Table 8-1	Non-printable ASCII characters	230
Table 9-1	[Job] Section Options.....	252
Table 9-2	[FormsAuth] Section Options	252
Table 9-3	[Crawlers] Section Options	252
Table 9-4	[Indexers] Section Options	252
Table 9-5	[Network] Section Options	252
Table 9-6	[Schedule] Section Options	253
Table 9-7	[Proxy] Section Options.....	253
Table 9-8	[DocRouter] Section Options	253
Table 9-9	[Criterion] Section Options	253
Figure 9-1	Distributed K2 Spider system candidate for document routing	279
Figure 9-2	Workflow of K2 Spider evaluating criteria.....	314

Preface

This guide is for administrators whose job it is to create and maintain Verity collections with the command-line indexers.

This preface contains the following sections:

- [Using This Book](#)
- [Related Documentation](#)
- [Verity Technical Support](#)

Using This Book

This section briefly describes the organization of this book and the stylistic conventions it uses.

Version

The information in this book is current as of K2 Enterprise version 6.0. The content was last modified March 15, 2005. Corrections or updates to this information may be available through the Verity Customer Support site; see [“Verity Technical Support” on page 21](#).

Organization of This Book

This book includes the following parts, chapters and appendixes:

- **Part I: Overview**
 - [Chapter 1, “Verity Indexers Overview”](#) describes the various Verity command-line indexing tools.
- **Part II: Verity Spider**
 - [Chapter 2, “Verity Spider Overview”](#) introduces Verity Spider.
 - [Chapter 3, “Verity Spider Reference”](#) describes the options for the Verity Spider command-line tool, `vspider`.
 - [Chapter 4, “Verity Spider Examples”](#) contains usage examples.
- **Part III: Verity K2 Spider**
 - [Chapter 5, “K2 Spider Overview”](#) introduces features of the K2 Spider.
 - [Chapter 6, “Building a K2 Spider System”](#) describes some of the ways you can distribute the K2 Spider Server components to build a K2 Spider system.
 - [Chapter 7, “Running K2 Spider Server”](#) describes how to run the K2 Spider Server (`k2spider_srv`).
 - [Chapter 8, “Using K2 Spider Client”](#) describes how to run the K2 Spider Client (`k2spider_cli`).
 - [Chapter 9, “Job Initialization File Reference”](#) describes the job initialization file and includes a sample file, `myjob.ini`.
 - [Chapter 10, “K2 Spider Examples”](#) contains usage examples.

Stylistic Conventions

The following stylistic conventions are used in this book.

Convention	Usage
Plain	Narrative text.
Bold	User-interface elements in narrative text: ■ Click Cancel to halt the operation.
<i>Italics</i>	Book titles and new terms: ■ For more information, see the <i>Verity K2 Getting Started Guide</i>. ■ An <i>index</i> is a Verity collection, parametric index, or recommendation index.
Monospace	File names, paths, and code: ■ The name .ext file is installed in: C:\Verity\Data\
<i>Monospace italic</i>	Replaceable strings in file paths and code: ■ user <i>username</i>
Monospace bold	Data types and required user input: ■ SrvConnect A connection handle. ■ In the User Interface text box, type user1.

The following command-line syntax conventions are used in this book.

Convention	Usage
[optional]	Brackets describe optional syntax, as in [-create] to specify a non-required option.
	Bars indicate “either or” choices, as in [option1] [option2] In this example, you must choose between option1 and option2.
{ required }	Braces describe required syntax in which you have a choice and that at least one choice is required, as in { [option1] [option2] } In this example, you must choose option1, option2, or both options.
required	Absence of braces or brackets indicates required syntax in which there is no choice; you must enter the required syntax element.
<i>variable</i>	Italics specify variables to be replaced by actual values, as in -merge <i>filename1</i>
...	Ellipses indicate repetition of the same pattern, as in -merge <i>filename1</i> , <i>filename2</i> [, <i>filename3</i> ...] where the ellipses specify , <i>filename4</i> , and so on.

Use of punctuation—such as single and double quotes, commas, periods—indicates actual syntax; it is not part of the syntax definition.

Related Documentation

The following guides provide more details on collections, regarding their creation, configuration and maintenance:

- *Verity Collection Reference*
- *Verity K2 Dashboard Administrator Guide*
- *Verity Locale Configuration Guide*
- *Verity Intelligent Classification Guide*
- Gateway guides: *Verity K2 Lotus Notes Gateway Guide*, *Verity K2 ODBC Gateway Guide*, *Verity K2 Documentum Gateway Guide*, *Verity K2 Exchange Gateway Guide*.

Verity Technical Support

Verity Technical Support exists to provide you with prompt and accurate resolutions to difficulties relating to using Verity software products. You can contact Technical Support using any of the following methods:

Telephone: (403) 294-1107

Fax: (403) 750-4100

Email: tech-support@verity.com

Web: <http://www.verity.com>

Product documentation, release notes, and document updates are available on the Verity Customer Support Site, at

<https://customers.verity.com>

It is recommended that you periodically check the Customer Support site for the existence of updates to this and other Verity product documents.

Access to the contents of the Customer Support site requires a user name and password. To obtain a user name and password, follow the signup instructions on the Customer Support site home page. You will need to supply your Verity entity ID and Verity license key.

PART I

Overview

- Chapter 1: Verity Indexers Overview

Verity Indexers Overview

This chapter introduces you to the Verity command-line indexers and describes some of their differences.

This chapter includes the following sections:

- [Verity Spider](#)
- [Verity K2 Spider](#)
- [The mkvdk Command-Line Tool](#)

Verity Spider

The Verity Spider is a single process indexer suitable for those who do not require the distributed and parallel processing capabilities of K2 Spider. The Verity Spider command-line tool, `vspider`, is configured through a series of options that can be entered at the command-line or saved in command files for easy re-use.

The Verity Spider is ideal for modest indexing, though it can also handle larger loads. If you need more flexibility with distributing the indexing workload, you should consider deploying K2 Spider which allows you to adjust your environment as needed.

There are no APIs to directly mimic the functionality of the Verity Spider.

Verity K2 Spider

K2 Spider is a service-based, distributed client/server indexing solution suitable for those with parallel processing capabilities or very large document sets. The K2 Spider consists of two command-line tools, the K2 Spider Server, `k2spider_srv(.exe)` and the K2 Spider Client, `k2spider_cli(.exe)`. A K2 Spider indexing system can be expanded as your need for more indexing power increases. The `k2spider_srv` tool can be run as a Controller, Crawler, Indexer individually or in various combinations.

A Controller is a K2 Spider Server process executed with the `-controller` option. The Controller is the heart of a K2 Spider system, managing Crawlers and Indexers and the gathering and distributing of indexing job information. K2 Spider Client requests, including indexing jobs and their accompanying initialization files, are passed to the Controller which then decides how to manage them.

A Crawler is a K2 Spider Server process executed with the `-crawler` option. An Indexer is a K2 Spider Server process executed with the `-indexer` option.

K2 Spider can also be used through the K2 Dashboard administration interface. This interface handles many of the complexities inherent in a distributed architecture. Although you are not able to exploit to the fullest extent the distributed nature of the K2 Spider components, you are still able to take advantage of the robustness and features of K2 Spider. For more information, see the *Verity K2 Dashboard Administrator Guide*.

The mkvdk Command-Line Tool

The `mkvdk` command-line tool is suitable for those with extremely precise indexing needs because you supply your list of documents to be indexed in a text file.

You can also use the `mkvdk` command-line tool in conjunction with Verity Spider or K2 Spider to perform housekeeping and ongoing optimization of your Verity collections. Such optimizations can go beyond the built-in capabilities of Verity Spider and K2 Spider.

For more information on `mkvdk`, see *Verity Collection Reference*.

PART II

Verity Spider

- [Chapter 2: Verity Spider Overview](#)
- [Chapter 3: Verity Spider Reference](#)
- [Chapter 4: Verity Spider Examples](#)

Verity Spider Overview

This chapter introduces you to the Verity Spider and describes how to use some of its key features.

This chapter includes the following sections:

- [Verity Spider Features](#)
- [How Verity Spider Indexes](#)

Verity Spider Features

The Verity Spider enables you to index documents in a variety of repositories throughout the enterprise. Verity Spider works in conjunction with Verity's KeyView document filtering technology so that more than two hundred of the most popular application document formats can be indexed, including Microsoft Office and WordPerfect, ASCII text, HTML, SGML, XML and PDF (Adobe Acrobat) documents.

State Maintenance Through a Persistent Store

Verity Spider stores the state of gathered and indexed URLs and documents in a persistent store, allowing it to track progress for the purposes of gracefully and efficiently restarting halted indexing jobs and intelligently refreshing changed documents.

The information in the persistent store can help you stay informed about such items as the number of indexed pages, number of visited pages, number of rejected pages, and number of broken links.

The command-line tool, `vsdb`, is available for you to interact with the persistent store, to manage document records and also obtain information and statistics. For more information, see ["Using vsdb" on page 94](#).

Warning! The Verity Spider persistent store is platform and version dependent.

- Verity Spider sv incorporates a persistent store format that is not compatible with previous versions.
- If you want to copy a 6.0 collection from one operating system platform to another, you will have to perform a synchronization, using the `vsdb` tool with the `-recreate` option. For more information, see ["Using vsdb" on page 94](#).

Performance

With low memory requirements, flow control and the help of multithreading and efficient Domain Name System (DNS) lookups, spidering performance is greatly improved over previous versions.

Multiple Connection and Indexing Threads

The Verity Spider separates the gathering and indexing work into multiple threads for concurrence. Verity Spider can create multiple concurrent connections to data repositories for fetching documents, and run multiple concurrent indexing threads for maximum utilization.

Automatic Collection Optimization

When an indexing job is done, so long as the `-nooptimize` option is **not** used, the Verity Spider automatically performs optimization work on the collection to prepare it for searching. By default, the actions of `maxmerge` and `vdbopt` are performed.

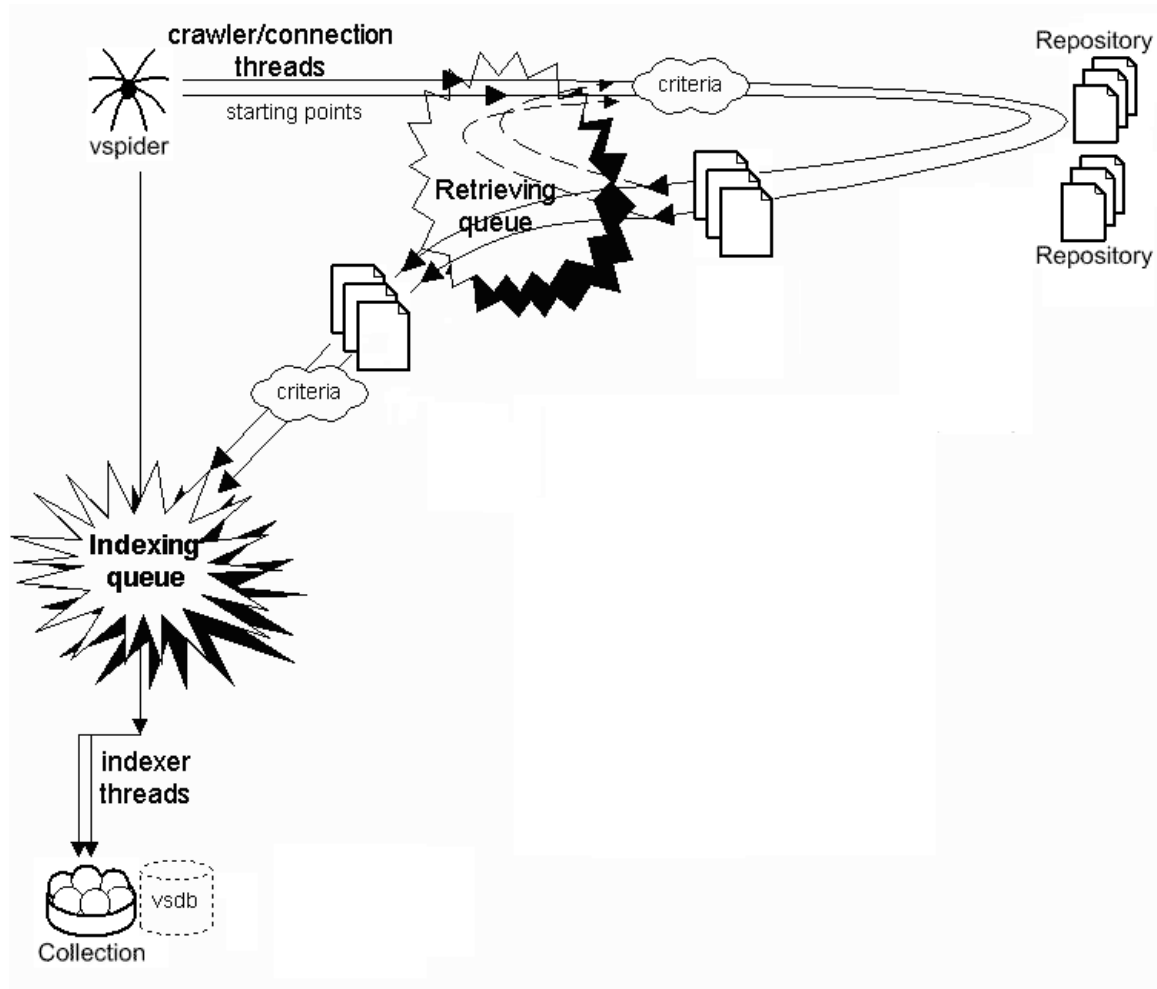
- The `maxmerge` action merges collection partitions to create partitions that are as large as possible. Each partition can have up to 64000 documents in it.
- The `vdbopt` action optimizes the internal Verity databases to make them as compact and streamlines as possible.

Note If you use the `-nooptimize` option, you should eventually use the `mkvdk` command-line tool to manually optimize the collection. No collection should remain unoptimized for long because search performance could degrade. The `mkvdk` command-line tool is described in the *Verity Collection Reference*. The `maxmerge` and `vdbopt` actions are available with `mkvdk`, along with other optimization and housekeeping options.

How Verity Spider Indexes

This section describes how the Verity Spider works.

Figure 2-1 Verity Spider indexing in action



1. When `vspider` is first run from the command-line, the following occurs:
 - ❑ The Retrieving and Indexing queues are readied
 - ❑ Crawler and indexer threads are started

The number of crawler threads can be specified using the `-connections` option.
The number of indexer threads can be specified using the `-indexers` option.
 - ❑ Criteria are loaded

Criteria include all options that affect what can be crawled and indexed, including `-include`, `-indskip`, `-cgiok` and so on.
2. Starting points (specified with `-start` or loaded from a gateway configuration file) are converted to document keys and then placed in the Retrieving queue.
3. The crawling threads pick up their work from the Retrieving queue, and evaluate the keys against the criteria.
4. The crawling threads retrieve document keys from the specified repositories.

Remember, starting points work in conjunction with the specified criteria to determine what is retrieved as candidates for indexing.
5. Documents are analyzed for links to new or updated documents.

For example, HTML documents are parsed for links to other documents, directories are walked for more files.
6. Any new work found is evaluated against the criteria and any documents that pass are placed in the Retrieving queue.

This essentially means that the work is passed back up to step 3.
7. The work picked up during step 3, both initially and with new work resulting from step 6, is evaluated against the criteria.

Any documents that pass the criteria for indexing are placed in the Indexing queue.
Steps 3 through 7 are repeated until the Retrieving queue is empty and all work is waiting in the Indexing queue.

8. The indexing threads pick up their work from the Indexing queue.

At this point, `vspider` builds bulk insert files (BIFs) and submits them to its internal Verity engine for indexing.

Note If you have specified `-processbif`, `-noindex` or `-nosubmit`, the following occurs:

- With `-processbif`, `vspider` processes what is specified as the BIFs are created.
- With `-noindex`, `vspider` creates the BIFs and submits them for indexing, but the indexing does not occur immediately.
- With `-nosubmit`, `vspider` creates the BIFs and does not submit them for indexing.

You must copy the BIFs from their work directory and perform whatever processing you want on them before eventually submitting them to the collection with `mkvdk`. If you do not copy the BIFs from their work directory, `vspider` will delete them as part of its optimization routine.

For more information on these options, see their descriptions in [Chapter 3, “Verity Spider Reference.”](#)

For more information on the `mkvdk` command-line tool, see the *Verity Collection Reference*.

9. When the Retrieving and Indexing queues are empty and the crawling and indexing threads are idle, `vspider` completes its optimization routines and then exits, returning to the command prompt.

Verity Spider Reference

The Verity Spider is a flexible and versatile indexing tool. The numerous options are provided here in an easy to use reference format.

This chapter includes the following sections:

- Verity Spider Syntax
- Reference of Command-Line Options
- Using vsdb
- Evaluating “include” and “exclude” Criteria
- The Use of Last-Modified Date

Verity Spider Syntax

The syntax for several basic types of Verity Spider indexing tasks is shown below.

Overview

Before you create an indexing task for a new collection, you should make copies of the relevant style files to ensure that you have a set of style files in a known, stable state.

Keep in mind that running multiple simultaneous Verity Spider jobs on the K2 Server host may cause performance problems for searches. This does not mean you should never run indexing jobs when users may be searching, because your collections are available for searching even while indexing jobs are running. With an eye toward optimizing performance, you should try staggering your indexing jobs to avoid overloading the host that has indexes and a search server running on it.

The Verity Spider Command

At its most basic level, a Verity Spider command consists of the following:

```
vspider -initialize -collection coll [options]
```

where *-initialize* is one of *-start* or *-refresh* (when starting points have changed), and *-collection* is required to provide a target for the Verity Spider, and *[options]* can be a near limitless combination of the options described later in this chapter.

There are of course dependencies for other options, depending on the nature of the indexing task. Some examples are:

- To build a new collection, you must use *-style*.
- To use a gateway other than File System or HTTP, you must specify the appropriate gateway option (*-notes* *-odbc* *-documentum* *-exchange*).
- To control how Verity Spider operates, including what documents it indexes, you should use at least some Verity Spider options.

If you do not run the Verity Spider executable from its installation directory, you must include that directory in your path. This is because the Verity Spider executable depends on other files to run properly.

The default location for the Verity Spider executable is as follows:

platformDir/bin

where *platformDir* is the directory in which you installed K2 Spider, and *platform* will vary depending on your operating system.

Using a Command File

If you want simpler reuse and archiving of your indexing commands, you should take advantage of the abstraction offered by the `-cmdfile` option. By using an ASCII text file to store a task's options, you also avoid the pitfall of using special characters in an option's parameter value.

For example, the `-processbif` option requires the use of "!" and therefore any task using that option must also use the `-cmdfile` option.

Indexing Job Scope

In order to index beyond the same, local host on which Verity Spider is running, you must have the appropriate license.

Reference of Command-Line Options

The following tables identify the Verity Spider options, grouped by type. Option names are case-sensitive.

Initialization Options

These options are used to initialize the `vspider` command-line tool and have something to do with how `vspider` functions throughout the session.

-start	-refresh	-refreshtime	-reparse
-restart			

Core Options.

These options are required by `vspider` to perform specific, special functions.

-cmdfile	-collection	-help	-jobpath
-style			

Gateway Options

These options are required by `vspider` to index the specific repository with the relevant gateway.

-notes -odbc	-gateway
-documentum	
-exchange	

Processing Options

These options concern how `vspider` processes documents for indexing.

-abspath	-checkacl	-detectdupfile	-indexers
-license	-maxfilehandles	-maxindmem	-maxnumdoc
-mimemap	-nodupdetect	-noindex	-nosubmit
-numpages	-persist	-preferred	-prefixmap
-processbif	-regexp	-submitsize	-temp

Networking Options

These options concern identification, performance and pathways when `vspider` accesses documents across the network.

-agentname	-connections	-delay	-header
-hostcache	-noflowctrl	-noproxy	-proxy
-proxyauth	-retry	-timeout	

Paths and URLs Options

These options concern what `vspider` is able to access at the path and URL levels.

-anonymous	-auth	-auth2	-certfile
-cgiok	-domain	-followdup	-followsymlink
-host	-https	-jumps	-keyfile
-nodocrobo	-nofollow	-norobo	-pathlen
-unlimited	-useget	-virtualhost	

Content Options

These options concern what `vspider` is able to access at a document level.

<code>-casesen</code>	<code>-exclude</code>	<code>-include</code>	<code>-indexclude</code>
<code>-indinclude</code>	<code>-indmimeexclude</code>	<code>-indmimeinclude</code>	<code>-indskip</code>
<code>-maxdocsize</code>	<code>-metafile</code>	<code>-mimeexclude</code>	<code>-mimeinclude</code>
<code>-mindocsize</code>	<code>-query</code>	<code>-skip</code>	

Locale Options

These options concern languages and related special processing.

<code>-charmap</code>	<code>-common</code>	<code>-datefmt</code>	<code>-keyencode</code>
<code>-language</code>	<code>-locale</code>	<code>-msgdb</code>	<code>-repos_charmap</code>

Logging Options

These options and their arguments concern what `vspider` logs while it is running.

<code>-loglevel</code>	<code>-debug -trace -verbose</code>	<code>-debug -trace -verbose</code>	<code>-debug -trace -verbose</code>

Maintenance Options

These options concern maintenance tasks `vspider` can perform.

<code>-nooptimize</code>	<code>-purge</code>	<code>-repair</code>	

Initialization Options

-start

Description A starting point for an indexing job.

Details You may list multiple values for `-start` by separating each one with a single space. You may alternatively list multiple `-start` options.

When you execute an indexing job from a command-line and you do not use a command file (with `-cmdfile`), you must URL-escape any special characters in the starting point. To URL-escape a special character, use "%hex-ASCII-character-number" in place of the character. For example, use `/time%26/` instead of `/time&/`. This enables the operating system to properly process the command string.

In the event an indexing task halts, you can re-run the task as-is. The persistent store for the specified collection is read and only those candidate URLs that are in the queue but not yet processed are parsed. Candidate URLs correspond to URLs of the following status as reported by `vsdb`:

`cand, used, inse, upda, dele, fail`

The `vsdb` command-line tool is available for you to interact with the persistent store. For more information about `vsdb`, see [“Using vsdb” on page 94](#).

Note By using `-start` with `-refresh`, you provide a starting point for Verity Spider and therefore do not need to use at least one of `-host`, `-domain`, `-nofollow` or `-unlimited`. You do not always need to explicitly include the `-refresh` option as Verity Spider automatically runs in a refresh mode when an indexing job is run again. However, to properly remove no longer valid records from file system indexing jobs, you should include the `-refresh` option.

Gateway Type is Web (HTTP Gateway)

For web sites, the starting point is the URL or URLs from which the Verity Spider is to begin crawling. Use other options such as `-jumps` to control how far from the starting point Verity Spider goes. URLs that are of the type `file://` are followed by `vspider`, but file system security is not supported, nor are links followed and parsed.

WARNING! The starting point for an indexing job against a secure web site must not contain a file name in the URL. Including the file name results in errors when trying to view documents returned from a search.

Note When indexing web sites on UNIX, if an HTML page contains a file URL of the type `file:///` that points to the root directory, Verity Spider follows that URL and indexes its content. You may want to use `-exclude "file://[/]"` in your Verity Spider command.

Gateway Type is File (File System Gateway)

For file systems, the starting point is the starting directory or directories in which the Verity Spider will start crawling. All subdirectories beneath the starting point will be crawled unless you use `-pathlen`, or any of the inclusion or exclusion criteria.

For information on indexing mapped network drives and UNC paths in Windows, see [“Indexing Network, UNC and Secure Paths on Windows” on page 144](#).

Gateway Type is Notes (Lotus Notes Gateway)

For Notes Servers, use the `-notes` option. When refreshing a collection, you can use `-start` with a Notes database to act as a filter on what is specified in the Lotus Notes Gateway configuration file. You cannot specify an individual document with the `-start` option when it is used with the `-notes` option.

For information about indexing with the Notes gateway, see the *Verity K2 Lotus Notes Gateway Guide*.

Gateway Type is ODBC (ODBC Gateway)

For ODBC servers, use the `-odbc` option. When refreshing a collection, you can use `-start` with a database to act as a filter on what is specified in the ODBC Gateway configuration file.

For information about indexing with the ODBC gateway, see the *Verity K2 ODBC Gateway Guide*.

Gateway Type is Documentum (Documentum Gateway)

For Documentum servers, use the `-documentum` option. When refreshing a collection, you can use `-start` with a Documentum folderkey to act as a filter on what is specified in the Documentum Gateway configuration file. You cannot specify an individual document with the `-start` option when it is used with the `-documentum` option.

For information about indexing with the Documentum gateway, see the *Verity K2 Documentum Gateway Guide*.

Gateway Type is Exchange (Exchange Gateway)

For Exchange Servers, use the `-exchange` option. When refreshing a collection, you can use `-start` with a Exchange folder to act as a filter on what is specified in the Exchange Gateway configuration file. You cannot specify an individual document with the `-start` option when it is used with the `-exchange` option.

For information about indexing with the Exchange gateway, see the *Verity K2 Exchange Gateway Guide*.

-refresh

Description Specifies that Verity Spider refresh a collection when starting points have changed.

Details When you re-run an existing indexing job for a collection indexed with any gateway other than the File System gateway, Verity Spider automatically refreshes the collection. However, there may be times when you need to manually specify the `-refresh` option. See the **Warning!** below for more details.

The `-refresh` option specifies that Verity Spider process only those documents which qualify as follows:

- They are new documents in the repository, and they qualify for indexing under the criteria.
- They exist in the collection and are recorded in the Verity Spider persistent store with a status of done. If Verity Spider determines that these indexed documents have been updated in the repository, then they are retrieved again to be reparsed and reindexed. Note that the document VdkVgwKey values do not change.
- They are deleted in the collection. If Verity Spider determines that documents have been deleted from the repository, then they are also deleted from the persistent store and the collection. The exception to this rule is when you use `-nooptimize` with `-refresh`. In this case, any document deleted from the repository is marked for deletion in the collection. It will be removed from the collection and the persistent store when the next indexing task is run for the collection.

Note You can also use `-start` to provide a starting point for Verity Spider. If you do not use `-start`, then you should use at least one of `-host`, `-domain`, or `-nofollow` or `-unlimited`. For further control, also see `-refreshtime`. If you do not use any constraint criteria, Verity Spider will operate without limits and will likely index far more than you desire.

Warning! When you re-run an existing indexing job for a collection with indexed file system starting points, Verity Spider does not automatically refresh the collection. This is done to avoid deleting documents that may be unavailable due to such things as network congestion and changes to access rights. Such causes may be temporary and you may not want documents affected by those cause removed from the collection.

If you add or remove any of the files or directories that were indexed from a file system, they will remain in a collection until you manually include the `-refresh` option in a subsequent indexing task. You must also use the `-start` option and refer to the parent directory from where the documents have been deleted.

-refreshtime

Description Specifies that any documents which have been indexed since the *timeunits* value began are not to be refreshed.

Syntax `-refreshtime timeunits`

Details The syntax for *timeunits* is: *n* day *n* hour *n* min *n* sec, where *n* is a positive integer. Note that there must be spaces, and since the first three letters of each time unit is parsed, you can use the singular or plural form.

Default All documents are refreshed.

Example If you specify:

```
-refreshtime 1 day 6 hours
```

then only those documents which were last indexed at least 30 hours and 1 second ago, will be refreshed.

Note This option is valid only with the `-refresh` option. When you use `vsdb -recreate`, the last indexed date is cleared.

-reparse

Description Forces parsing of all HTML documents already in the collection.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details You must specify a starting point with the `-start` option when you use `-reparse`.

You can use `-reparse` when you want to include paths and documents which were previously skipped due to exclusion or inclusion criteria. Remember to change the criteria, else there will be little for the Verity Spider to do. This can be easy to overlook when you are using `-cmdfile`.

Default No value.

-restart

Description Specifies that after a halt, `vspider` is to read the persistent store for the specified collection and process the candidate records that are in the work queue.

Details Candidate records are those that are of any of the following status types:

`cand, used, inse, upda, dele, fail`

To determine which records are of the above status types, use `vsdb`, described later in this chapter in “Using `vsdb`.”

Warning! This option should not be used after any indexing jobs in which you specified options that do not update the persistent store (such as `-noindex` or `-nosubmit`).

Note When you specify `-restart`, you cannot use `-start` or `-refresh`, but you must specify a collection with `-collection`. When you specify `-restart` with a collection that contains web sites (indexed with the HTTP gateway), you must also specify at least one of `-host`, `-domain`, `-nofollow` or `-unlimited`. When you specify `-restart` with a collection that contains file systems (indexed with the File System gateway), the default behavior is to limit `vspider` to the host of the starting points.

Core Options

-cmdfile

Description Specifies that Verity Spider reads command-line syntax from a file in addition to the options passed in the command-line.

Syntax `-cmdfile path_and_filename`

Details This option includes the path name to the file containing the options. The `-cmdfile` option circumvents command-line length limits.

The syntax for the command-file is:

option parameters option2 parameters2 optionN parametersN

For better readability, you should put each option and any parameters on a single line. Verity Spider will be able to properly parse the lines.

Example A sample command-file is:

```
-start c:\mypath  
-start d:\mypath2  
-abspath  
-style c:\stylepath  
-collection c:\collpath\collname  
-jobpath c:\jobpath
```

Note It is highly recommended you take advantage of the abstraction offered by this option. User error in erroneously including or omitting options in subsequent indexing jobs can be greatly reduced.

The paths provided in the Example above are for illustration purposes only. Make sure your paths are valid and accurate. Also, your jobs may require more options. Remember to include everything you need for your job in the command file.

-collection

Description The full path to the collection you want to create or update.

Syntax `-collection path_and_filename`

Warning! You will receive an error if you specify a filename with an extension of `.clm`. Support for meta collections was discontinued as of Verity Spider V3.5.

-help

Description Displays the Verity Spider options.

-jobpath

Description Specifies the location of the Verity Spider databases and the indexing job-related files and directories.

Syntax `-jobpath path`

Details The job-related directories and their contents are:

- `log` — All Verity Spider log files. See `-loglevel` for descriptions of the log files.
- `bif` — Bulk insert files.
- `temp` — Web pages cached for indexing.

You can also specify the temp directory by using the `-temp` option.

- `admin` — Files created by `vspider`.

These directories are created for you beneath the last directory specified in *path*.

Default If you do not use `-jobpath`, Verity Spider will create a `/spider` directory within the collection. For multiple-collection tasks, the first collection specified will be used.

Note You must make sure that *path* values are unique for all indexing jobs.

When using `-purge`, specify the correct job path with `-jobpath` so that the persistent store is handled properly.

Warning! You cannot use multiple job paths for multiple simultaneous indexing tasks for the same collection. Only one indexing task at a time can run for a given collection.

-style

Description Specifies the path to the style files to use when creating a new collection.

Syntax `-style path`

Details When specifying `-style`, you must use the style files appropriate for each indexing gateway. Style files created with the K2 Dashboard are located in:

`dataDir/stylesets/stylesetName`

where `dataDir` is the data directory.

For information on the K2 Dashboard and creating style sets, see the *Verity K2 Dashboard Administrator Guide* and the relevant gateway guide if necessary.

You can create style sets for the Lotus Notes Gateway, the Documentum Gateway, and the Exchange Gateway using the StyleSet Editor in standalone mode. See the relevant gateway guide for more information on where the style sets can be created.

Note You can safely omit `-style` when resubmitting an indexing job as the style information will already be part of the collection. If you are using `-cmdfile`, you can leave it there.

Gateway Options

-notes -odbc -documentum -exchange

Description The type of repository you are going to index.

Type Required in order to use one of the corresponding Verity gateways.

Details You must use the `-style` option to specify the style files appropriate for a particular indexing gateway. For more information, see `-style`.

See Also `-anonymous` for use with `-notes` and `-query` for use with `-odbc`.

For more information on configuring the files necessary for Verity Spider to index a particular repository, use the following table.

For this repository...	See this documentation...
file systems, web servers	<i>Verity K2 Dashboard Administrator Guide</i>
Notes servers	<i>Verity K2 Lotus Notes Gateway Guide</i>
ODBC servers	<i>Verity K2 ODBC Gateway Guide</i>
Documentum servers	<i>Verity K2 Documentum Gateway Guide</i>
Exchange servers	<i>Verity K2 Exchange Gateway Guide</i>

-gateway

Description Signifies that a custom gateway is being used.

Type For use by OEMs of Verity software only.

Processing Options

-abspath

Description Generates absolute paths for files.

Type File system only. This option can only be used with the File System gateway.

Details Use this option when the document locations are not going to change, but the collection might be moved around.

When you index a web server's contents through the file system, you should use [-prefixmap](#) with `-abspath` to map the absolute file paths to URLs.

For more information, see [“Prefix Mapping” on page 147](#).

Default Absolute paths are not generated. By default, Verity Spider stores paths relative to the location of the collection into which documents are being indexed. You must specify the `-abspath` option to store absolute paths.

See Also [-prefixmap](#)

-checkacl

Description Enables `vspider` to update documents where the access time has changed.

Details Use this option to update documents where the access times have changed, for example, when they are opened and edited.

Note The access time for a document changes for a variety of reasons. Specifying this option may result in more documents being indexed than you desired.

-detectdupfile

Description Enables checksum-based detection of duplicates when indexing file systems using the File System gateway.

Type File system only. This option can only be used with the File System gateway.

Details By default, duplicate detection is disabled when indexing documents using the File System gateway. By using `-detectdupfile`, a checksum is computed based on the CRC-32 algorithm. The checksum combined with the document size is used to determine if the document is a duplicate.

Default Verity Spider does not detect duplicate files when indexing file systems using the File System gateway.

Note By default, duplicate detection is enabled for Web Site indexing. You can disable it by using the `-nodupdetect` option.

-indexers

Description Specifies the maximum number of indexing threads to run on a collection.

Syntax `-indexers num_indexers`

Details Increasing the value for `-indexers` requires additional CPU and memory resources.

Default 2 indexing threads.

See Also `-maxindmem`

-license

Description Specifies the license file to use.

Syntax `-license path_and_filename`

Default By default, `ind.lic` is used, from:

`installDir/k2/common/`

where `installDir` is the directory in which you installed K2 Spider.

-maxfilehandles

Description When running Verity Spider on UNIX, specifies the maximum number of file handles of memory, in kilobytes, to be used by each indexing thread.

Syntax `-maxfilehandles n`

Details The number of file handles per VDK session started by `vspider`.

Default The default value is $(ulimit -256)/(\text{number of indexer sessions})$.

Note Use this option with the help of Verity Technical Support.

-maxindmem

Description Specifies the maximum amount of memory, in kilobytes, used by each indexing thread

Syntax `-maxindmem kilobytes`

Details The number of threads is specified with `-indexers`.

Default Each indexing thread uses as much memory as is available from the system.

-maxnumdoc

Description Specifies the maximum number of documents to be downloaded or submitted for indexing.

Syntax `-maxnumdoc num_docs`

Details The value for *num_docs* does not necessarily correspond exactly to the number of documents indexed. The following factors affect the actual number.

- Whether or not the value of *num_docs* falls within a block of documents dictated by `-submitsize`. If it does, the entire block of documents must be processed.
- Whether or not documents retrieved are actually indexed due to being corrupt.
- Whether or not the document is new and has not been inserted into the collection yet. The *num_docs* value applies only to new, inserted documents. When you refresh an indexing job, previously indexed documents will not count against the value. This includes changed documents, because although those documents will be indexed, they are already inserted in the collection. Only new documents that are newly inserted count against the value of *num_docs*.

Default There is no maximum number of documents.

-mimemap

Description Specifies a control file (simple ASCII text) that maps file extensions to MIME-types.

Syntax `-mimemap path_and_filename`

Details This option enables you to make custom associations and override defaults.

The format for the control file is:

<code>#file_ext_no_dot</code>	<code>mime-type</code>
<code>abc</code>	<code>application/word</code>

Default No file is specified.

See Also [“Prefix Mapping” on page 147](#).

-nodupdetect

Description Disables checksum-based detection of duplicates when indexing web sites using the HTTP gateway.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details With this option, URL-based duplicate detection is still performed.

By default, duplicate detection is enabled when indexing documents using the HTTP gateway. A document checksum is computed based on the CRC-32 algorithm. The checksum combined with the document size is used to determine if the document is a duplicate. By using `-nodupdetect`, you can disable duplicate detection when indexing documents using the HTTP gateway.

Default Verity Spider detects duplicates based on a checksum when indexing Web sites using the HTTP gateway.

See Also [-followdup](#)

-noindex

Description Specifies that the Verity Spider gathers document locations, without indexing them.

Details Gathered document locations are stored in a bulk insert file (BIF), which is then submitted to the collection. This option must be used in conjunction with the alternative indexing process, `mkvdk`. You cannot re-run `vspider` because as part of its housekeeping duties, `vspider` cleans out the working directory that contains the BIFs created with the `-noindex` option. After running an indexing job with `vspider` with which you specified the `-noindex` option, you need to move the BIFs to another directory so they can be used with `mkvdk`.

Do not try to start both the Verity Spider and another process at the same time. You must allow Verity Spider enough time to generate enough work for the secondary indexing process to act upon. If you are using `mkvdk`, you can run it in persistent mode to ensure it will act upon work generated by Verity Spider.

Warning! When you execute an indexing job for a collection and you use `-noindex`, the persistent store for the collection is not updated.

See Also `-nosubmit` for a similar and alternative means of creating a BIF for use with `mkvdk`.

For more information on `mkvdk`, see the *Verity Collection Reference*.

-nosubmit

Description Specifies that the Verity Spider gathers document locations without indexing them.

Details Gathered document locations are stored in a bulk insert file (BIF), which is **not** submitted to the collection. This option must be used in conjunction with the alternative indexing process, `mkvdk`. You cannot re-run `vspider` because as part of its housekeeping duties, `vspider` cleans out the working directory that contains the BIFs created with the `-noindex` option. After running an indexing job with `vspider` with which you specified the `-noindex` option, you need to move the BIFs to another directory so they can be used with `mkvdk`.

You must specify the name and path for the BIF with `mkvdk` because the collection has no record of the location of the BIF.

-numpages

Description Specifies the maximum number of memory pages to be used by each indexing thread.

Syntax `-numpages n`

Details The number of memory pages per VDK session started by `vspider`.

Note Use this option with the help of Verity Technical Support.

-persist

Description Enables the Verity Spider to run in persistent mode

Syntax `-persist num_seconds`

Details When running in persistent mode, the Verity Spider checks for updates every *num_seconds* seconds until it is stopped.

While the Verity Spider is running in persistent mode, there is no optimization. Once the Verity Spider is taken out of persistent mode, you will need to perform optimization on the collection. For more information about using `mkvdk` for optimization, see the *Verity Collection Reference*.

Warning! You should not run more than one Verity Spider process in persistent mode. As the Verity Spider is a resource intensive process, you should only run it in persistent mode with an interval of less than one day. For time intervals greater than twelve hours, you should use some form of scheduling. Some examples are `cron` jobs for UNIX, and the `AT` command for Windows.

-preferred

Description Specifies a list of hosts or domains which are to be preferred when retrieving documents for viewing.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-preferred expression_1 [expression_n] ...`

You may list multiple values for `-preferred` by separating each one with a single space. You may alternatively list multiple `-preferred` options.

Details You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. To use regular expressions with this option, you must also specify `-regex` option. Use this option when you leave duplicate detection enabled and do not specify `-nodupdetect`.

When indexing, you may encounter a non-preferred host first. In that case, documents are parsed and followed and stored as candidates. When duplicates are encountered on another server, which is preferred, the duplicate documents from the non-preferred server are skipped. When documents are requested for viewing, they will be retrieved from the preferred server.

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

See Also `-regex`

-prefixmap

Description Specifies a control file (simple ASCII text) that maps one field to another.

Syntax `-prefixmap path_and_filename`

Details This option is typically used to create a URL field that is the Web equivalent of a file system path. File system indexing is faster than web crawling over the network. Using `-prefixmap` to replace the file system path with the URL means relative hyperlinks in the HTML pages are kept intact when viewed through K2 Server.

A scenario is you want to index from a web server is limited to the files available by way of a file system rather than through the web server itself.

The format for the control file is:

```
src_field src_prefix dest_field dest_prefix
```

Default No file path is specified.

See Also `-abspath`, and [“Using the Control File” on page 148](#).

-processbif

Description Specifies a command string in which you call a program or script that operates on Bulk Insert Files (BIFs) generated by Verity Spider.

Syntax `-processbif "command_string !*"`

Details Due to the use of special characters which represent the BIF, you must run Verity Spider with a command file using the `-cmdfile` option.

For example, to use a script called `fix_bif` to add customized information to BIF files, use the following command:

```
vspider -cmdfile filename
```

where *filename* is the text-only command file which contains at least:

```
-processbif "fix_bif !*"
```

Note Enclose paths that contain spaces with double quotes (`"`) that are escaped with a backslash (`\`). For example:

```
-processbif "perl \"E:\colls\process_bif\hello.pl\" \"!*\")"
```

The string `!*` is a placeholder for the BIF path. As it may contain spaces, it should also be enclosed in escaped double quotes as shown.

-regexp

Description Specifies the use of regular expressions rather than the default wild-card expressions.

Details This option affects the following options: `-exclude`, `-indexexclude`, `-include`, `-indinclude`, `-skip`, `-indskip`, `-preferred`, and `-nofollow`.

Wild-card expressions allow the use of the asterisk (`*`) for text strings, and the question mark (`?`) for single characters.

This wild-card expression...	Will apply to these text strings...
<code>a*t</code>	although, attitude, audit
<code>file?.htm</code>	files.htm, file1.htm, filer.htm
<code>name?.*</code>	names.txt, name.doc, named.blank, names.ext

Regular expressions allow for more powerful means for matching alphanumeric strings.

For example, to match "ab11" or "ab34" but not "abcd" or "ab11cd," you could use the following regular expression:

```
^ab[0-9][0-9]$
```

Default Only wild-card expressions can be used.

-submitsize

Description Specifies the number of documents submitted for indexing at one time.

Syntax `-submitsize num_documents`

Details The upper limit is 64,000.

Although larger values mean more efficient processing by the indexer, smaller values will allow more parallelism on multi-CPU systems. Furthermore, in the event of a halt during indexing, a smaller value means fewer documents will be lost.

If a halt occurs during indexing, the chunk of documents specified by `-submitsize` is lost because there is no transactional rollback for indexing and the documents are no longer in the queue for indexing. Remember that when you re-run the indexing task, Verity Spider can only continue with URLs and documents which are enqueued.

Default 1024 documents.

-temp

Description Specifies the directory for temporary files (disk cache).

Syntax `-temp path`

Default By default, the temp directory is contained within the job directory (optionally specified with the `-jobpath` option).

If you do not specify a value for this option, Verity Spider will create a `/spider/temp` directory within the collection. For multiple-collection tasks, the first collection specified will be used.

Note Make sure the location you specify contains enough disk space to handle the documents which are downloaded and held before indexing. The documents are deleted from the hard disk after they are indexed.

See Also `-jobpath`, for specifying the location of all indexing job directories and files, one of which is the temp directory.

Networking Options

-agentname

Description Specifies the value for the agent name field that is part of the HTTP request.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-agentname string`

Details Since web servers can be configured to return different versions of the same page depending on the requesting agent, you can use `-agentname` to impersonate a browser client.

Use double-quotes if the name contains a space. Use `-cmdfile` if the agent name you want to use contains forbidden characters such as slashes or backslashes.

Default No value.

-connections

Description Specifies the maximum number of simultaneous socket connections to make to web sites for indexing.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-connections num_connections`

Details Each socket connection to a web site implies a separate thread. Valid values are 1 to 100.

Default 6 connections.

Note Verity Spider's dynamic flow control makes the most use of all available connections when indexing web sites. If you are indexing multiple sites, you may want to increase this number. Note that increasing the number of connections may not always help because of such dependencies as your network connection and the capabilities of the remote hosts.

See Also `-noflowctrl`

-delay

Description Specifies the minimum time between HTTP requests in milliseconds.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-delay num_milliseconds`

Default 0 milliseconds for no delay.

-header

Description Specifies an HTTP header to be added to the spidering request.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-header string`

Details An example HTTP header to send is:

```
-header "Referer: http://www.verity.com/"
```

Verity Spider sends some predefined headers, such as Accept and User-Agent among others, by default. Special headers are sometimes necessary to correctly index a site.

For example, versions of Verity Spider prior to 4.5 did not support the "Host" header, which is needed for Virtual Host indexing. Also, a "Proxy-authentication" header was needed to pass a username and password to a proxy server.

The "Host" header is supported by default, and the `-proxyauth` option is available for proxy server authentication. Therefore the `-header` option is maintained only for backwards compatibility and possible future enhancements.

Default No value.

Note Misuse of this option will cause spider failure. In the event that this happens, re-run the indexing task with modified `-header` values.

-hostcache

Description Specifies the number of hostnames to cache to avoid DNS lookups.

Syntax `-hostcache num_hostnames`

Default The default value is 256.

-noflowctrl

Description Disables round-robin indexing of web sites with network flow control.

Type Web crawling only. This option can only be used with the HTTP gateway.

Default When indexing web sites, Verity Spider distributes requests to web servers in a round-robin manner. This means one URL is fetched from each web server in turn. With flow control, it is possible that a faster web site will finish before a slower one. Regardless, the Verity Spider optimizes indexing on every web server.

Verity Spider adjusts the number of connections per server depending on the download bandwidth. When the download bandwidth from a web server falls below a certain value, Verity Spider will automatically scale back the number of connections to that web server. There will always be at least one connection to a web server. When the download bandwidth increases to an acceptable level, Verity Spider reallocates connections (per the value of the `-connections` option).

Warning! When using `-noflowctrl`, you may see a significant drop in performance.

See Also `-connections`

-noproxy

Description Specifies that the Verity Spider directly access the hosts whose names match those specified.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-noproxy name_1 [name_n] ...`

Details By default, when `-proxy` is specified, the Verity Spider first tries to access every host with the proxy information. To improve performance, also use `-noproxy` for those hosts you know can be accessed without a proxy host. For the *name* variable, you can use the asterisk (`*`) wild card for text strings. For example:

```
'*.verity.com'
```

You cannot use the question mark (`?`) wild card, nor can you use regular expressions with this option even if you have specified the `-regexp` option.

On Windows, you should include double quotes around the argument to protect the special character (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

Default No value.

Note You must have valid Verity Spider licensing capability for remote indexing for this option to be useful.

For information on how to configure an HTTP gateway configuration file so documents can be retrieved for viewing, see [“Indexing with Proxy Servers” on page 140](#).

-proxy

Description Specifies host and port for proxy server.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-proxy proxyhost:port`

Default No value.

Note You must have valid Verity Spider licensing capability for remote indexing for this option to be useful.

For information on how to configure an HTTP gateway configuration file so documents can be retrieved for viewing, see [“Indexing with Proxy Servers” on page 140](#).

See Also [-proxyauth](#) for proxy servers that require authentication, and [-noproxy](#) for hosts which you know are accessible without having to go through a proxy server.

-proxyauth

Description Specifies login information for proxy server connections that require authorization to get outside the firewall.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-proxyauth username:password`

Details This option is used in conjunction with [-proxy](#).

Default No value.

Note You must have valid Verity Spider licensing capability for remote indexing for this option to be useful.

For information on how to configure an HTTP gateway configuration file so documents can be retrieved for viewing, see [“Indexing with Proxy Servers” on page 140](#).

-retry

Description Specifies the number of times the Verity Spider should attempt to access a URL.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-retry num_retries`

Details You should use `-retry` when it is likely that an unstable network connection will give false rejections.

Default 4 retries.

-timeout

Description Specifies the time period, in seconds, that the Verity Spider should wait before timing out on a network connection and on accessing data.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-timeout num_seconds`

Details The data access value is automatically twice the value you specify for the network connection timeout.

Default The network connection timeout is 30 seconds, with the value for the data access timeout being 60 seconds.

Paths and URLs Options

-anonymous

Description Specifies that the Verity Spider index publicly accessible Notes databases as an anonymous user.

Type Indexing Notes databases only. This option can only be used with the Notes gateway.

Details Use this option with the `-notes` option. When you use `-anonymous`, you will not be able to index any private Notes databases.

-auth

Description Specifies an authorization file to support authentication for secure paths.

Syntax `-auth path_and_filename`

Details For more information, see [“Indexing Secure Web Sites” on page 134](#).

Note If you are using `vspider` with a style set generated by the StyleSet Editor, you must use the `-auth2` option. The authorization file generated by the StyleSet Editor, `vgwhttp.auth`, contains an encrypted password for which you must use `-auth2` to decrypt it.

If you are creating your own authorization file in which you store passwords as plain text, you can use `-auth`.

-auth2

Description Specifies the authorization file, *vgwgateway.auth*, that is generated by the StyleSet Editor, where *gateway* can be one of *http*, *fsys*, *odbc*, *dctm*, *notes* or *exch*.

Syntax `-auth2 path_and_filename`

Details If you are using *vspider* with a style set generated by the StyleSet Editor, you must use the `-auth2` option. The *vgwgateway.auth* file contains an encrypted password and you must use `-auth2` instead of `-auth` to allow *vspider* to decrypt it.

If you are creating your own authorization file in which you store passwords as plain text, you can use `-auth`.

See Also For more information, see [“Indexing Secure Web Sites” on page 134](#).

-certfile

Description Specifies the path to the SSL client certificate to be used for indexing SSL-secured web sites which require client certificates.

Type Web crawling only. This option can only be used with the HTTP gateway. This option is required with `-keyfile`.

Syntax `-certfile path_and_filename`

Details The format of the certificate must be in the form of an X.509 certificate.

See Also `-https` and `-keyfile`

-cgiok

Description enables indexing of URLs containing the ? symbol.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details URLs which contain the ? symbol typically lead to a CGI or other such processing program.

The return document produced by the web server is indexed and parsed for document links which are followed and in turn indexed and parsed. However, if the web server does not return a page, perhaps because the URL is missing parameters which are required for processing in order to produce a page, then nothing happens. There is no page to index and parse.

Default Verity Spider cannot index URLs containing the ? symbol.

Example A URL without parameters is: `http://server.com/cgi-bin/program?`

If you include parameters in the URL to be indexed, as specified with the `-start` option, then those parameters are processed and any resulting pages are indexed and parsed.

By default, URLs with ? symbols are skipped.

-domain

Description Limits indexing to the specified domain(s).

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-domain name_1 [name_n] ...`

You may list multiple values for `-domain` by separating each one with a single space. You may alternatively list multiple `-domain` options.

Details You must use only complete text strings for domains. You may not use wild-card expressions. URLs not in the specified domain(s) will not be downloaded or parsed.

Default Verity Spider is limited to the host on which it is running.

-followdup

Description Specifies that Verity Spider follows links within duplicate documents, although only the first instance of any duplicate documents will be indexed.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details You may find this option useful if you use the same home page on multiple sites. By default, only the first instance of the document is indexed, while subsequent instances are skipped. If you have different secondary documents on the different sites, using `-followdup` will allow you to get to them for indexing, while still indexing the common home page only once.

Default Verity Spider does not follow links within duplicate documents.

-followsymlink

Description Specifies that Verity Spider follows symbolic links when indexing UNIX file systems.

Type File system only. This option can only be used with the File System gateway.

Default Verity Spider does not follow symbolic links.

-host

Description Limits indexing to the specified host or hosts.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-host name_1 [name_n] ...`

You may list multiple values for `-host` by separating each one with a single space. You may alternatively list multiple `-host` options.

Details You must use only complete text strings for hosts. You may not use wild-card expressions.

Default Verity Spider is limited to the host specified in the values for `start`.

-https

Description enables the indexing of SSL-enabled web sites.

Type Web crawling only. This option can only be used with the HTTP gateway and in conjunction with the SSL Option that is separately available and licensed.

Default Verity Spider cannot index SSL-enabled Web sites.

Note You must have the Verity SSL Option installed to use `-https`. The Verity SSL Option is a Verity Spider add-on available separately from a Verity salesperson.

See Also *Verity SSL Option Release Notes*

-jumps

Description Specifies the maximum number of levels deep an indexing job can go from the starting URL.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-jumps num_jumps`

Details Specify a number between 0 and 254. If you see extremely large numbers of documents in a collection where you do not expect them, you should consider experimenting with this option, in conjunction with the Content options, to pare down your collection.

Default There is no limit on the number of jumps.

-keyfile

Description Specifies the path to the private key file to be used with `-certfile`.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-keyfile path_and_filename`

See Also `-certfile` and `-https`

-nodocrobo

Description Specifies ROBOT META tag directives are to be ignored.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details In HTML 3.0 and earlier, robot directives could only be given as the file `robots.txt` under the root directory of a web site. In HTML 4.0, every document can have robot directives embedded in the META field. Use this option to ignore them.

Default ROBOT META tag directives are honored.

See Also `-norobo` and <http://www.w3c.org/TR/REC-html40/html40.txt>.

-nofollow

Description Specifies Verity Spider cannot follow any links specified by the element `<a href="link">` tag when the *expression* occurs between `<a href="link">` and `</a>`.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-nofollow "expression"`

You may list multiple values for `-nofollow` by separating each one with a single space. You may alternatively list multiple `-nofollow` options.

Details The value for *expression* must match any of the text that lies between the `<a href="link">` and `</a>` HTML tags. For example, given `<a href="http://myweb/file.htm">term1 term2</a>`, use the following:

```
-nofollow term1
```

The link to `http://myweb/file.htm` is not followed.

If you do not specify a value for *expression* and instead only specify `-nofollow`, then Verity Spider assumes a value of `"*"` where no links are followed.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. You should always encapsulate the *expression* values in double quotes to ensure they are properly interpreted.

If you use backslashes, you must double them so they are properly escaped. For example:

```
C:\\test\\docs\\path
```

To use regular expressions with this option, you must also specify `-regex` option.

Default Verity Spider follows all links in HTML documents.

-norobo

Description Specifies that any `robots.txt` files encountered are ignored.

Type Web crawling only. This option can only be used with the HTTP gateway.

Details The `robots.txt` file is used on many web sites to specify what parts of the site indexers should avoid. The default is to honor any `robots.txt` files.

If you are re-indexing a site and `robots.txt` has changed, the Verity Spider will delete documents that have been newly disallowed by `robots.txt`.

Default `robots.txt` files are honored.

See Also [-nodocrobo](#) and <http://info.webcrawler.com/mak/projects/robots/norobots.html>.

-pathlen

Description Limits indexing to the specified number of path segments in the URL or file system path.

Syntax `-pathlen num_pathsegments`

Details The path length is determined as follows:

- The host name and drive letter are not included. For example, neither `www.spider.com:80/` nor `C:\` are included in determining the path length.
- All elements following the host name are included.
- The actual file name, if present, is included. For example, `/world.html` is included in determining the path length.

Any directory paths between the host and the actual file name are included.

Example For the following URL, the path length is 4:

```
http://www.spider:80/comics/fun/funny/world.html
      <-1->  <2>  <-3->  <---4--->
```

For the following file system path, the path length is 3:

```
C:\files\docs\datasheets
      <-1->  <-2->  <---3--->
```

Default 100 path segments.

-unlimited

Description Specifies no limits to be placed on Verity Spider if neither `-host` nor `-domain` is specified.

Default Verity Spider is limited to the host of the first starting point.

-useget

Description Specifies that only the HTTP GET method is used to retrieve web pages.

Type Web crawling only. This option can only be used with the HTTP gateway.

Default Verity Spider attempts to use the HEAD method first, then the GET method.

-virtualhost

Description Specifies that DNS lookups are avoided for the hosts listed.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-virtualhost name_1 [name_n] ...`

You may list multiple values for `-virtualhost` by separating each one with a single space. You may alternatively list multiple `-virtualhost` options.

Details You must use only complete text strings for hosts. You may not use wild-card expressions. This enables you to index by alias, such as when multiple web servers are running on the same host. You can use regular expressions.

Normally, when Verity Spider resolves host names, it uses DNS lookups to convert the names to canonical names, of which there can be only one per machine. This enables the detection of duplicate documents, to prevent results from being diluted. In the case of multiple aliased hosts, however, duplication is not a barrier as documents can be referred to by more than one alias, and yet remain distinct because of the different alias names.

Example You may have both `marketing.verity.com` and `sales.verity.com` running on the same host. Each alias has a different document root, although both may have `index.htm` files. With `-virtualhost`, both server aliases can be indexed as distinct sites. Without `-virtualhost`, they are both be resolved to the same host name and only the first document encountered from any duplicate pair is indexed.

Warning! If you are using Netscape Enterprise Server, and you have specified only the host name as a virtual host, then Verity Spider will not be able to index the virtual host site. This is because the Verity Spider always adds the domain name to the document key.

Content Options

-casesen

Description Makes processing document keys case-sensitive. Use only for indexing UNIX servers. By default, processing is not case-sensitive.

Details Keep in mind that when you use this option you may need to specify multiple criteria entries. For example, `-exclude *.css` will only exclude files ending with `.css` and not files ending with `.CSS`.

-exclude

Description Specifies that files, paths and URLs matching the specified expression(s) will not be followed. By default, nothing is explicitly excluded.

Syntax `-exclude exp_1 [exp_n] ...`

You may list multiple values for `-exclude` by separating each one with a single space. You may alternatively list multiple `-exclude` options.

Details If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
'/my_doc*/year199?'
```

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

To use regular expressions with this option, you must also specify `-regex` option.

To specify a file, path or URL which you want followed but not indexed, use `-indexexclude`. For document types, use `-mimeexclude` instead. For example, specify `-mimeexclude application/pdf` rather than `-exclude *.pdf`.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `-exclude`.

See Also [“Evaluating “include” and “exclude” Criteria” on page 104.](#)

-include

Description Specifies that only those files, paths and URLs which match the specified expression or expressions will be followed.

Syntax `-include exp_1 [exp_n] ...`

You may list multiple values for `-include` by separating each one with a single space. You may alternatively list multiple `-include` options.

Details If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
'/my_doc*/year199?'
```

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

To use regular expressions with this option, you must also specify `-regex` option.

Keep in mind that if your starting points do not contain the specified `-include` expressions, nothing will be indexed. The `-include` option prevents Verity Spider from even following anything which does not match the specified expressions. You may want to use `-indinclude` instead. Where `-include` prevents Verity Spider from even following anything which does **not** match the specified expressions, `-indinclude` enables Verity Spider to follow what matches the specified expressions, while not indexing.

For document types, use `-mimeinclude` instead. For example, specify `-mimeinclude text/html` rather than `include *.htm`.

Default Nothing is explicitly included.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `-include`.

See Also [“Evaluating “include” and “exclude” Criteria” on page 104.](#)

-indexclude

Description Specifies that the files and paths in URLs which match the expressions are not indexed.

Type Web crawling and file system indexing only. This option can only be used with the File System and HTTP gateways.

Syntax `-indexclude exp_1 [exp_n] ...`

You may list multiple values for `-indexclude` by separating each one with a single space. You may alternatively list multiple `-indexclude` options.

Details Files and paths in URLs are still followed. If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
'/my_doc*/year199?'
```

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

To use regular expressions with this option, you must also specify `-regexp` option.

Use this option to gather some documents, such as HTML tables of contents, to gain access to other documents for indexing.

Where the `-exclude` option prevents Verity Spider from even following anything which matches the specified expressions, `-indexclude` enables Verity Spider to follow anything while only skipping that which matches the specified expressions.

For document types, use `-indmimeexclude` instead.

Default Nothing is explicitly excluded from being indexed.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `-indexclude`.

See Also [“Evaluating “include” and “exclude” Criteria” on page 104.](#)

-indinclude

Description Specifies that only those files and paths in URLs which match the expressions be followed and indexed.

Type Web crawling and file system indexing only. This option can only be used with the File System and HTTP gateways.

Syntax `-indinclude exp_1 [exp_n] ...`

You may list multiple values for `-indinclude` by separating each one with a single space. You may alternatively list multiple `-indinclude` options.

Details If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
'/my_doc*/year199?'
```

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

To use regular expressions with this option, you must also specify `-regex` option.

Where the `-include` option prevents Verity Spider from even following anything which does not match the specified expressions, `-indinclude` enables Verity Spider to follow anything while only indexing that which matches the specified expressions.

Default Nothing is explicitly included for indexing.

Example If you want to index all documents that include “search” in the URL at `http://web.verity.com`, you cannot use:

```
vspider -collection collname -start http://web.verity.com -include '*search*'
```

This is because the starting point does not match the `-include` criteria. Instead, use `-indinclude` to follow all documents (unless, of course, you have specified any of the exclude options) and index only those documents that match your criteria. Simply replace `-include` with `-indinclude` in the above example.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `-indinclude`.

See Also “Evaluating “include” and “exclude” Criteria” on page 104.

-indmimeexclude

Description Specifies that only those MIME types which match the expressions be followed but not indexed.

Type Web crawling and file system indexing only. This option can only be used with the File System and HTTP gateways.

Syntax `-indmimeexclude mime_1 [mime_n] ...`

You may list multiple values for `-indmimeexclude` by separating each one with a single space. You may alternatively list multiple `-indmimeexclude` options.

Details On Windows, you should include double quotes around the argument to protect the special characters such as (*). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

Use this option to gather some documents, such as HTML tables of contents, to gain access to other documents for indexing. The `-mimeexclude` option, on the other hand, prevents specified documents from being followed at all. For the `mime` variable, you can include the asterisk (`*`) wild card for text strings. For example:

```
"text/htm*"
```

You cannot use the question mark (`?`) wild card, nor can you use regular expressions with this option even if you have specified the `-regex` option.

Default No MIME Types are explicitly excluded from being indexed.

See Also ["Setting MIME Types" on page 152.](#)

-indmimeinclude

Description Specifies that only those MIME types which match the expressions be followed and indexed.

Type Web crawling and file system indexing only. This option can only be used with the File System and HTTP gateways.

Syntax `-indmimeinclude mime_1 [mime_n] ...`

You may list multiple values for `-indmimeinclude` by separating each one with a single space. You may alternatively list multiple `-indmimeinclude` options.

Details You should use `-indmimeinclude` instead of `-mimeinclude` because `-mimeinclude` does not allow you to index desired documents if the starting URL is not followed. With `-indmimeinclude`, crawling will be able to happen, but only documents that match the specified MIME Type will be indexed.

For the *mime* variable, you can include the asterisk (`*`) wild card for text strings. For example:

```
"text/htm*"
```

On Windows, you should include double quotes around the argument to protect the special character (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

You cannot use the question mark (`?`) wild card, nor can you use regular expressions with this option even if you have specified the `-regexp` option.

Default No MIME Types are explicitly included for indexing.

Example If you want to index all Word documents at `http://web.verity.com`, you cannot use:

```
vspider -collection collname -style style_dir -start http://  
web.verity.com -mimeinclude 'application/msword'
```

This is because the starting point does not match the `-mimeinclude` criteria. Now, you can use `-indmimeinclude` to follow all documents (unless, of course, you have specified any of the exclude options) and index only those documents that match your criteria. Simply replace `-mimeinclude` with `-indmimeinclude` in the above example.

See Also [“Setting MIME Types” on page 152.](#)

-indskip

Description Specifies Verity Spider is to follow and parse links within, but not index, any HTML document that contains the text of *expression* between the starting and ending *HTML_tag*.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-indskip HTML_tag "expression"`

You may list multiple values for `-indskip` by separating each one with a single space. You may alternatively list multiple `-indskip` options.

Details For multiple *HTML_tag* and *expression* combinations, use multiple instances of the `-indskip` option.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example: a `'/my_doc*/year199?'`.

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`. To use regular expressions with this option, you must also specify `-regexp`.

Default No HTML documents are skipped from indexing.

Example To skip all HTML documents which contain the word "personnel" in the Title element, while still parsing those documents for links, use `-indskip title "personnel"`. For any document that contains the word `personnel` between `<title>` and `</title>`, the content of the document itself will not be indexed, but links on the page will be parsed.

Example To avoid indexing directory listing pages, while still parsing the document and path links except for link up to the parent directory, use one of the following depending on the Web server being indexed:

For Netscape Web servers, use the following:

```
-indskip title "*Index of*"
-nofollow "*parent directory*"
```

For Microsoft Internet Information Server, use the following:

```
-indskip a "*to parent directory*"
-nofollow "*parent directory*"
```

-maxdocsize

Description Specifies the maximum size, in kilobytes, for documents to be indexed.

Syntax `-maxdocsize integer`

Details Any documents larger than the value specified by `maxdocsize` will be ignored. During crawling, documents that exceed the threshold are dropped as they are streamed.

Default 20MB. If you are indexing .PST or .ZIP files, this value may be low. You may need to increase the value to account for larger file sizes.

-metafile

Description Enables you to use a text file to map custom meta tags to valid HTTP header fields.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-metafile path_and_filename`

Details If you use backslashes, you must double them so they are properly escaped. For example: `C:\\test\\docs\\path`.

This means you are able to use your own meta tag, in the document, to replace what is returned by the web server, or to insert it if nothing is returned. Currently, the only header fields of real value are "Last-Modified" and "Content-Length."

The syntax for the two possible entries in the text file is:

```
name Last-Modified y/n
name Content-Length y/n
```

where *y/n* is an override flag which can be either yes or no.

Default No file is specified.

Example A mapping file for `-metafile` might include:

```
Doc_Last_Touched Last-Modified n
Doc_Size Content-Length y
```

If you use the *y* override flag, the value for the custom meta tag overrides the value for the valid field, even if both values are present and differ. This can be useful when the valid field value is always sent, but you want to specify your own value with a custom meta tag.

If you use the *n* override flag, then the value for the custom meta tag will be used only if there is no value for the valid field returned by the server. If a value for the valid field exists, then that is given precedence.

Warning! If you have several entries mapping to the same valid field, only the last entry will take effect.

See Also [“The Use of Last-Modified Date” on page 111.](#)

-mimeexclude

Description Specifies MIME types which are neither followed nor indexed.

Syntax `-mimeexclude mime_1 [mime_n] ...`

You may list multiple values for `-mimeexclude` by separating each one with a single space. You may alternatively list multiple `-mimeexclude` options.

Details On Windows, you should include double quotes around the argument to protect the special characters such as (*). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

The default is to include all known MIME types. For the *mime* variable, you can include the asterisk (*) wild card for text strings. For example:

```
"text/*"
```

You cannot use the question mark (?) wild card, nor can you use regular expressions with this option even if you have specified the `-regexp` option.

Use `-indmimeexclude` to allow the Verity Spider to follow documents, without indexing them, to gain access to other desirable document types.

Default No MIME Types are explicitly excluded.

See Also [“Setting MIME Types” on page 152.](#)

-mimeinclude

Description Specifies MIME types to be included.

Syntax `-mimeinclude mime_1 [mime_n] ...`

You may list multiple values for `-mimeinclude` by separating each one with a single space. You may alternatively list multiple `-mimeinclude` options.

Details On Windows, you should include double quotes around the argument to protect the special characters such as (*). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

The default is to include all known MIME types. For the *mime* variable, you can include the asterisk (`*`) wild card for text strings. For example:

```
"text/*"
```

You cannot use the question mark (`?`) wild card, nor can you use regular expressions with this option even if you have specified the `-regex` option.

Default No MIME Types are explicitly included.

See Also [“Setting MIME Types” on page 152.](#)

-mindocsize

Description Specifies the minimum size, in kilobytes, for documents to be indexed.

Syntax `-mindocsize integer`

Details Any documents smaller than the value specified by `mindocsize` will be ignored.

Default There is no limit on document size.

-query

Description Specifies a SQL query to determine what to index.

Type This option can only be used with the ODBC gateway.

Syntax `-query "your_WHERE_clause"`

Default Verity Spider will index everything allowed according to the criteria in the `vgwodbc.cfg` configuration file.

See Also See the documentation for your database server for more information on SQL syntax. Also see the *Verity K2 ODBC Gateway Guide*.

Example Assuming you want to index a table, `Table1`, with `pkCol1` as the ODBC gateway primary key (specified as `primarykey:pkCol1` in `vgwodbc.cfg`), include the following in your `vspider` command:

```
vspider -odbc -style stylepath  
        -collection odbc_coll -query "Col1<1000 AND Col1>100"
```

The value for `-query` will be translated by the ODBC gateway into the following SQL query when crawling and gathering document keys:

```
SELECT pkCol1 FROM Table1 WHERE Col1<1000 AND Col1>100
```

The `-query` option is useful only in the following situations:

- ❑ You are indexing only one repository (DSN) and one table in that repository
- ❑ You use a single `-start` value with which the `-query` option will be associated

-skip

Description Specifies Verity Spider is to not index any HTML document that contains the text of *expression* between the starting and ending *HTML_tag*.

Type Web crawling only. This option can only be used with the HTTP gateway.

Syntax `-skip HTML_tag "expression"`

You may list multiple values for `-skip` by separating each one with a single space. You may alternatively list multiple `-skip` options.

Details For multiple *HTML_tag* and *expression* combinations, use multiple instances of the `-skip` option.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
-skip a '/my_doc*/year199?'
```

On Windows, you should include double quotes around the argument to protect the special characters such as (`*`). On UNIX, you should use single quotes. Note that this is only required when you run the indexing job from a command line. Quotes are not necessary within a command file (`-cmdfile`).

To use regular expressions with this option, you must also specify `-regexp` option.

Default All HTML documents are indexed.

Example To skip all HTML documents that contain the word "personnel" in the Title element, use the following:

```
-skip title "personnel"
```

Example To skip all HTML documents that contain both the word "private" and the phrase "internal user" in any paragraph element, use the following:

```
-skip title "personnel"  
-skip p "*internal use*"
```

Locale Options

-charmap

Description Specifies the character map to use for the specified locale.

Syntax `-charmap charset_name`

Details Identifies the character set of the files being passed with the indexing job. For example, the files for `-cmdfile`, `-mimemap`, `-prefixmap` and other options that pass in files. The *charset_name* must be the name of one of the supported character sets for the collection's locale.

See Also Appendix A of the *Verity Locale Configuration Guide* lists the supported character sets for each Verity locale and indicates which one is the internal character set.

-common

Description Specifies path to the Verity home directory, *installDir*/k2/common, where *installDir* is the directory in which you installed K2 Spider.

Default *installDir*/k2/common

Note This option is typically not needed, as long as the PATH environment variable is set correctly.

-datefmt

Description Specifies the Verity import date format to use. Valid values are MDY, DMY, YMD, USA and EUR.

Syntax `-datefmt format`

Default The default value is MDY.

-keyencode

Description Enables encoding of the DOC_FN field.

Note Use this option with the help of Verity Technical Support.

-language

Description Specifies the Verity locale to use in indexing.

Syntax `-language name`

Details This option is being replaced by the semantically consistent `-locale`, and is still supported for backwards compatibility.

-locale

Description Specifies the Verity locale to use when creating the collection.

Syntax `-locale locale_name`

Details This option is identical to `-language`. *locale_name* must be the name of a locale for which you are licensed, and must be one of the Verity locales listed in Appendix A of the *Verity Locale Configuration Guide*.

This option is not required if the collection uses the default session locale (usually `uni`).

Default The default value is installation-dependent.

See Also For more information on supported locales and their use, see the *Verity Locale Configuration Guide*.

-msgdb

Description Specifies the path to the `ind.msg` message database file.

Syntax `-msgdb path`

Details The `ind.msg` message database is read from:

`installDir/k2/common/ind.msg`

where `installDir` is the directory in which you installed K2 Spider.

-repos_charmap

Description Specifies the character map name of the operating system so that the file path in the `VdkVgwKey` field can be converted to the character map of the collection.

Type Web crawling and file system indexing only. This option can only be used with the File System and HTTP gateways.

Syntax `-repos_charmap oscharmapname`

Details The file path and name to be indexed may contain special characters and be in a particular character map (of the operating system). Use this option to have Verity Spider convert the character map of the file path and name (as it is stored in the `VdkVgwKey` field) from the operating system character map to the character map of the collection into which the file is being indexed.

Example You are indexing a file system where files contain SJIS characters in their names. Specify `-repos_charmap sjis` in your Verity Spider command.

Logging Options

-loglevel

Description Specifies the types of messages to log.

Syntax `-loglevel [nostdout] argument`

Details If you add `nostdout` to the `loglevel` argument, messages will not be written to standard output. Log files, however, will still be created.

Default Messages are written to standard output and to various log files in the subdirectory named `/log` beneath the Verity Spider job directory.

Message type	Description
information	Licensing information written to <code>info.log</code> . Included with all arguments.
warning	Warning messages written to <code>warning.log</code> . Included with all arguments.
error	Error messages written to <code>error.log</code> . Included with all arguments.
badkey	Messages regarding keys which could not be indexed due to invalid documents, written to <code>badkey.log</code> . Included with all arguments. Files of MIME Types unsupported for indexing, such as <code>.EXE</code> or <code>.JPG</code> , are reported as bad keys and written to <code>badkey.log</code> .
progress	Current state of a document key written to <code>progress.log</code> . Note that a key with a progress of "inserting" may wind up as a badkey and therefore skipped, rather than an indexed key. Included with all arguments.
summary	Inserted, indexed and ignored messages written to <code>summary.log</code> . Included with all arguments except <code>skip</code> .
skip	Skipped documents, with explanation, written to <code>skip.log</code> . Included with all arguments, except <code>summary</code> and <code>verbose</code> . Note that documents skipped due to URL redirection, where you do not specify the redirected URL as a candidate for indexing, are not recorded in <code>skip.log</code> .
debug	Internal Verity Spider processing messages such as <code>enqueued</code> , written to <code>debug.log</code> . Included with both <code>debug</code> and <code>trace</code> arguments.
trace	Internal Verity Spider processing messages written to <code>debug.log</code> . Included only with the <code>trace</code> argument.

Choose one of the following arguments to determine what message types are logged.

Loglevel arguments	Description
notrunc	Specifies that <code>vspider</code> continue to append to the existing log file without truncating it. By default, log files are truncated at 100MB and moved to a secondary log file, <code>logfile.name.old</code> . At most, you have access to the last 199MB of the log files unless you specify <code>notrunc</code> .
summary	Includes the following message types: information, warning, error, badkey, progress, summary Use this option only if you do not want skip type messages.
skip	Includes the following message types: information, warning, error, badkey, progress, skip Use this option only if you do not want summary type messages.
verbose	Includes the following message types: information, warning, error, badkey, progress, summary, skip
debug	Includes the following message types: information, warning, error, badkey, progress, summary, skip, debug This argument should be used only at the direction of Verity technical support or for troubleshooting indexing problems.
trace	Includes the following message types: information, warning, error, badkey, progress, summary, skip, debug This argument should be used only at the direction of Verity technical support or for troubleshooting indexing problems.

-debug -trace -verbose

Description These options are maintained for backwards compatibility. They specify the types of messages to log.

Maintenance Options

-nooptimize

Description Prevents the Verity Spider from optimizing the collection when the indexing task is completed.

Details Using this option can reduce processing overhead during the indexing job. You should use this option sparingly, as it leaves the collection in less than optimum shape. Some examples of when you might want to use this option are:

- You want to manually perform custom optimization of the collection, using `mkvdk`.
- You are running multiple indexing jobs against a collection, and want to wait until they are all finished to optimize.

Generally, you should not leave a collection unoptimized for too long, as search times can slow significantly.

In brief, optimizing a collection means creating a small number of large partitions, which can greatly reduce search times.

Default Verity Spider optimizes collections when indexing tasks finish. Verity Spider optimization mimics the `mkvdk` actions of `maxmerge` and `vdbopt`.

See Also The *Verity Collection Reference*.

-purge

Description Deletes document tables and index files in the collection, and cleans up the collection's persistent store.

Details The collection is then "fresh" with its original style files, and is not deleted from the file system.

Note The collection you are purging must be off-line. For information on taking collections off-line with K2 Dashboard, see "Taking a Collection Off-line" in the chapter titled "Managing Collections," of the *Verity K2 Dashboard Administrator Guide*. You must also make sure you specify the correct job path with `-jobpath` to ensure that the persistent store is handled properly.

-repair

Description Specifies a failure-recovery mode for the collection, where the goal is to determine the causes of any errors, repair the errors (if possible), and bring a collection back up.

Details Although the Verity indexing engine always leaves the collection in a consistent, usable state, and no data can be lost or corrupted due to machine failures, it *is* possible for a process or event *external* to the Verity engine to corrupt one or more collections. You can use `-repair` for constant failure-recovery operation, or you can run it selectively on collections that are "down."

Using vsdb

During an indexing task, document keys can be in any one of several processing stages. With the command-line tool, `vsdb`, you can generate a report on the fetched and indexed status of document keys in a collection's persistent store. The persistent store is the database in which Verity Spider stores information about what it is indexing and what has been indexed.

vsdb Arguments

The following table contains the supported arguments for `vsdb`, which is installed to:

`platformDir/bin`

where `platformDir` is the directory in which you installed K2 Services, and `platform` is dependent on your operating system. For a complete list of supported operating systems and platforms, see the *Verity K2 Release Notes*.

Note The `vsdb` command-line tool is for use with the Verity Spider only.

Argument	Description
<code>-anonymous</code>	Used with <code>-recreate</code> , this option specifies the use of the anonymous user to access the persistent store and the collection. Use this option only if you are recreating a Notes collection which was indexed using the Verity Spider option, <code>-anonymous</code> . For Notes collections which were created with credentials, use the <code>-usegw vsdb</code> option. For more information, see the <code>vspider</code> option " <code>-anonymous</code> " on page 67 .
<code>-casesen</code>	This argument specifies case-sensitivity for queries.
<code>-coll</code>	This argument prints all keys that have actually been indexed into the collection.
<code>-collection path</code>	This argument specifies the collection with which you want to interact. When you specify the collection name without any other arguments, you get a summary for the collection. The summary includes the total number of keys fetched, the number of keys in the collection, and counts for each status code. See the description for <code>-status</code> below for the status codes.

Argument	Description
-common	Used with -recreate, this argument specifies the location of the Verity common files which were used to create the collection. For more information, see the <code>vspider</code> option -common.
-compact	This argument removes records marked for deletion from the persistent store.
-convert	This argument converts Verity Spider V3.6 persistent stores to the latest format. Persistent stores for Verity Spider after V3.6 due not need converting.
-date	Prints the Last-Modified (<code>last_modified_date</code>) and last indexed dates for each key in the persistent store. The Last Modified date for a file is returned by the web server or file system. The last indexed date is retrieved from the persistent store and it represents the last time the key record was modified in the persistent store. For information on how Verity Spider uses the Last-Modified date of a document, see “The Use of Last-Modified Date” on page 111 .
-dateformat	Used with -recreate, this argument specifies the date format which was used to create the collection. For more information, see the <code>vspider</code> option “-datefmt” on page 87.
-delcoll <i>item</i>	This argument specifies the records to delete from the collection based on the supplied <i>item</i> . Note that you cannot specify multiple <i>items</i> , such as: <code>-delcoll ‘*html’ -delcoll ‘*pdf’</code> You may only use a single <i>item</i> with the -delcoll argument. To report on different -delcoll criteria, run multiple instances of <code>vsdb</code> . For Windows, use double-quotes.
-delete	This argument specifies the action of marking for deletion items from the persistent store, based on the supplied criteria with either -match, -key or -status. You must specify one of -match, -key or -status to qualify what is to be deleted. To completely remove the existence of all records marked for deletion, you will have to eventually run <code>vsdb</code> with the -compact argument.
-finddup	This argument, used in conjunction with -preferred, specifies that <code>vsdb</code> is to search for duplicate document entries in the persistent store. For more information on using -finddup to purge duplicate documents, see “Purging Duplicate Documents” on page 98 .
-jobpath	This argument specifies the path for the <code>vsdb</code> files. When used with -finddup, this argument specifies the paths to the two persistent stores that you want to compare in order to flag duplicate document records.
-key <i>key_name</i>	This argument finds the key with the exact key name specified. You can only match on a single exact key name at a time.

Argument	Description
-locale	Used with -recreate, this argument specifies the locale which was used when you created the collection. For more information, see the <code>vspider</code> option “-locale” on page 88 . For information on supported locales and their use, see the <i>Verity Locale Configuration Guide</i> .
-match <i>expression</i>	This argument prints keys that match the specified wild-card expression (<i>expression</i>). Note that you cannot use multiple expressions, such as: <pre>-match '*html' -match '*pdf'</pre> You may only use a single expression with the -match argument. To report on different -match criteria, run multiple instances of vsdb. For Windows, use double-quotes.
-parent	Prints the parent, or referring, key for the keys to be indexed. For example, if you are indexing <code>bar.html</code> , and <code>foo.html</code> contains a link to it, -parent will return <code>foo.html</code> as a parent to <code>bar.html</code> .
-preferred <i>expression</i>	This argument defines the URL to prefer when there is more than one instance of a document. Specify the URL, using an <i>expression</i>, from which the document instance you want to keep came. You can use the question mark (?) and asterisk (*) wildcards in the <i>expression</i>, where ? represents a single character, and * represents multiple characters. For example: <code>-preferred *www.verity.com*</code> If you do not use -preferred, the document instance that corresponds to the second path value for -jobpath is marked for deletion from that job's persistent store. To completely remove its existence, you will have to eventually run vsdb with the -compact argument. For more information on using -preferred to purge duplicate documents, see “Purging Duplicate Documents” on page 98. See also: -finddup, -delete, -delcoll and -jobpath.
-print	Prints keys in the persistent store for a collection. Use this option when you want to see the actual document keys for your query criteria.
-recreate	This argument specifies that the persistent store is to be recreated. This is necessary when you move collections to another operating system or you must recover from a corrupted persistent store. This functionality replaces the -resync option for Verity Spider. The -resync option for Verity Spider is no longer supported. WARNING! Once you have used -recreate on the persistent store for a collection, there are no historical contents for refresh jobs. See also: -locale, -common, -dateformat, -anonymous and -usegw.

Argument	Description																				
<code>-status status</code>	Prints keys matching the specified status code(s). Multiple status codes are combined with the Boolean AND operator. Choose from: <table><tr><td><code>none</code></td><td>key is not being processed</td></tr><tr><td><code>cand</code></td><td>key is a candidate for indexing</td></tr><tr><td><code>used</code></td><td>key is being fetched</td></tr><tr><td><code>inse</code></td><td>key is in inserting queue</td></tr><tr><td><code>upda</code></td><td>key is in update queue</td></tr><tr><td><code>dele</code></td><td>key is in delete queue</td></tr><tr><td><code>done</code></td><td>key has been processed</td></tr><tr><td><code>fail</code></td><td>key fetching has failed</td></tr><tr><td><code>dup</code></td><td>key is a duplicate of another URL</td></tr><tr><td><code>skip</code></td><td>key was skipped</td></tr></table> The lifecycle of a key in the persistent store is: <code>new key > cand > used > (dup or fail or skip or inse or upda or dele) > done</code> A successful indexing job will mean keys in either a none or done status. If you notice many keys in any of the other states, it is likely the indexing job was interrupted. You should restart the job so that keys with a status of cand, used, inse, upda, dele, or fail are processed.	<code>none</code>	key is not being processed	<code>cand</code>	key is a candidate for indexing	<code>used</code>	key is being fetched	<code>inse</code>	key is in inserting queue	<code>upda</code>	key is in update queue	<code>dele</code>	key is in delete queue	<code>done</code>	key has been processed	<code>fail</code>	key fetching has failed	<code>dup</code>	key is a duplicate of another URL	<code>skip</code>	key was skipped
<code>none</code>	key is not being processed																				
<code>cand</code>	key is a candidate for indexing																				
<code>used</code>	key is being fetched																				
<code>inse</code>	key is in inserting queue																				
<code>upda</code>	key is in update queue																				
<code>dele</code>	key is in delete queue																				
<code>done</code>	key has been processed																				
<code>fail</code>	key fetching has failed																				
<code>dup</code>	key is a duplicate of another URL																				
<code>skip</code>	key was skipped																				
<code>-usegw</code>	Used with <code>-recreate</code>, this option specifies that authentication is required to access the collection. Use this option only if you are recreating a Notes, Exchange, ODBC or Documentum collection. For Notes collections which were indexed using the Verity Spider option, <code>-anonymous</code>, use the <code>-anonymous vsdb</code> option. You must supply the same credentials you used when you created the collection.																				

vsdb Examples

This section contains examples of using vsdb to remove duplicate documents from collections, and to rebuild a corrupted persistent store.

- [Purging Duplicate Documents](#)
- [Removing Duplicate Documents from Searches](#)
- [Restoring a Corrupted Persistent Store](#)

Purging Duplicate Documents

When the same document is indexed into more than one collection, you may want to define a single instance of the document as preferred so that search users do not get duplicate results.

Two scenarios where you may index duplicate documents are:

- A document is indexed from one place into multiple collections by one or more vs spider jobs.
- The exact same document is indexed from multiple places into multiple collections by one or more vs spider jobs.

By defining a preferred host, and using the `-delcoll` and `-delete` vsdb options, only the document instance from the preferred host will be maintained in the collection and the persistent store. All other instances of the document will be purged from the collection and persistent store.

Example—Two Identical Documents Indexed from Two Hosts

You have a document (`doc1.txt`) that exists on two hosts (`hostA` and `hostB`), and is indexed into two different collections. You use `job1` to index `doc1.txt` from `hostA` into collection `collA`, and `job2` to index `doc1.txt` from `hostB` into collection `collB`. You know `hostA` is the faster system, so you want to define `hostA` as preferred for `doc1.txt`. Furthermore, you want the other instance of `doc1.txt`, from `hostB`, removed from the appropriate collection and persistent store.

To keep the document instance indexed from `hostA`, use the following command:

```
vsdb -jobpath path1 path2 -finddup -preferred *hostA*  
      -delete -delcoll -collection collB -match doc1.txt
```

In the vsdb command, which you issue from a command prompt, *path1* and *path2* are the paths to the persistent stores in the `jobpath` directories for `job1` and `job2`.

vsdb will compare the persistent store of *vspider job1* (specified by *path1*) with the persistent store of *vspider job2* (specified by *path2*).

Each item in the vsdb command example is described in the following table.

Option	Type/Description
vsdb	The Verity Spider command-line tool for interacting with persistent stores.
-jobpath	This option specifies the path for the vsdb files.
path1	This is the full path to the persistent store for job1. In the example, job1 indexes doc1.txt from hostA into collection coll1.
path2	This is the full path to the persistent store for job2. In the example, job2 indexes doc1.txt from hostB into collection coll2.
-finddup	This option, used in conjunction with -preferred, triggers the search for duplicate document entries.
-preferred *hostA*	This option specifies that hostA is the preferred repository for doc1.txt. If you do not use -preferred, the document instance with the older last-modified date is marked for deletion. If the last-modified date is the same, then the document that corresponds to the second path value for -jobpath is marked for deletion from that job's persistent store. To completely remove records marked for deletion, you will have to eventually run vsdb with the -compact argument.
-delete	This option specifies that the value for the -match option, doc1.txt, will be marked for deletion in the persistent store for job2. To completely remove its existence, you will have to eventually run vsdb with the -compact argument.
-delcoll	This option specifies that the value for the -match option, doc1.txt, will be marked for deletion in collB.
-collection collB	This option specifies the collection to be targeted by -delcoll.
-match doc1.txt	This option specifies the document to be targeted by -delete and -delcoll.

Example—One Document Indexed into Two Collections

You have a document (`text2.doc`) that exists on host `hostQ`, and is indexed into two different collections by two different indexing jobs. Assume the last-modified date for `text2.doc` remains the same for both indexing jobs. You use `job1` to index `text2.doc` into collection `coll1`, and `job2` to index the same `text2.doc` into collection `collB`.

When you discover that you have two instances of the same document in two different collections, you want to remove the instance indexed by `job2` into `collB`.

To keep the document instance indexed by `job1` into `collA`, use the following command:

```
vsdb -jobpath path1 path2 -finddup -delete -delcoll  
-collection collB -match text2.doc
```

Since you know you want to keep the instance from `job1` (that uses `path1`), and you know the last-modified date is the same for both document instances, you do not need to use the `-preferred` argument.

Removing Duplicate Documents from Searches

When a document exists on two hosts and is indexed into two different collections, you may want to remove one instance of the document so that search users do not get duplicate results.

Two scenarios where you may index duplicate documents are:

- The same document from the same place is indexed into two collections by two different `vspider` jobs.

For example, you could have indexed `doc1.txt` from `hostA` into both collection `coll1` and collection `coll2`.

- The same document from two different places is indexed into two collections by two different `vspider` jobs.

For example, you could have indexed `doc1.txt` from `hostA` into collection `coll1`, and an identical copy of `doc1.txt` from `hostB` into collection `coll2`.

To prevent users from getting duplicate results, you need to remove one of the document instances from one of the collections as well as the persistent store for the `vspider` job that indexed into that collection. To remove one of the documents so that only one is returned as a result, use a combination of the following `vsdb` options:

Option	Type/Description
<code>-jobpath</code>	specifies the path to the persistent stores for the two <code>vspider</code> jobs
<code>-finddup</code>	triggers the search for duplicate documents.
<code>-preferred</code>	establishes a document instance to keep, and by extension an instance to delete.
<code>-delete</code>	marks the non-preferred document for deletion from the persistent store.
<code>-delcoll</code>	marks the non-preferred document for deletion from the collection.

For example, to keep the document instance indexed from hostA rather than the duplicate indexed from hostB, issue the following vsdb command from the command-line:

```
vsdb -jobpath path1 path2 -finddup -preferred hostA -delete  
doc1.txt -delcoll doc1.txt
```

where *path1* is the path to the persistent store in the job directory for *vspider job1* that indexed *coll1*, and *path2* is the path to the persistent store in the job directory for *vspider job2* that indexed into *coll2*.

vsdb will compare the persistent store of *vspider job1* with the persistent store of *vspider job2*, found in *path2*.

The `-finddup` option causes vsdb to flag *doc1.txt* because it occurs in two places. The `-preferred` option causes vsdb to save the document instance that was indexed from *hostA*. Together, the `-delete` and the `-delcoll` options mark for deletion the instance of *doc1.txt* that was not indexed from *hostA*. In this case this means the instance of *doc1.txt* that was indexed from *hostB* by *vspider job2* will be marked for deletion from both *coll2* and the persistent store for *vspider job2*.

In the scenario where the same document from the same place is indexed into two collections from two *vspider* jobs, the document instance whose last-modified date is older will be marked for deletion. If the last-modified date is identical, then the document instance that corresponds to *path2* is marked for deletion.

Note If you do not specify `-preferred`, or no documents match the values you provide for `-preferred`, then the document instance that corresponds to *path2* is marked for deletion.

Restoring a Corrupted Persistent Store

There may be times when a collection's persistent store becomes corrupted. For more information on the persistent store, see [“State Maintenance Through a Persistent Store” on page 30](#).

To restore a corrupted persistent store, follow these steps:

1. Delete the `rwdb.lock` file in the job directory for an indexing job.

The job directory can be defined with the `-jobpath` option. If you do not use `-jobpath`, Verity Spider will create a `/spider/job` directory within the collection specified in the indexing job. For multiple-collection tasks, the first collection specified will be used.

2. Run the following `vsdb` command:

```
vsdb -jobpath job_path -collection coll_path -recreate
```

where *job_path* and *coll_path* are specific to the current situation.

WARNING! Once you have used `-recreate` on the persistent store for a collection, there are no historical contents for refresh jobs.

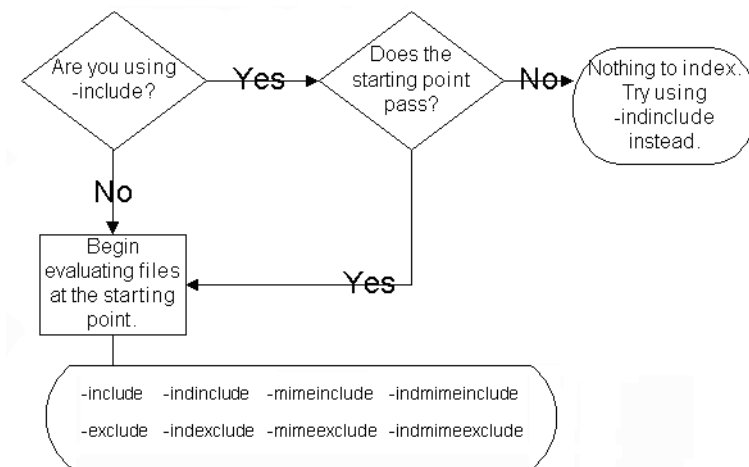
Evaluating “include” and “exclude” Criteria

When you specify any of the include or exclude criteria with a Verity Spider indexing job, all candidates (URL, path and file) are evaluated against all of the specified criteria. The evaluation occurs as a logical AND across the criteria options.

Evaluation Workflow

The basic workflow for evaluating criteria is illustrated in the following diagram. Verity Spider evaluates each criteria option in turn in an arbitrary order.

Figure 3-1 Workflow of Verity Spider evaluating criteria



The `-include` option concerns following and indexing. Therefore, when you use it in your indexing job, the starting point must pass to determine whether or not Verity Spider can proceed to any of the files at the starting point.

For example, if you specify `-include '*memo*'` and `-start 'http://web.verity.com/docs'`, nothing will happen. The starting point does not match the `-include` string.

Since `-indinclude` only concerns indexing while always making it possible to follow, you should use it instead of `-include` in most cases.

Example Indexing Job

The following example indexing job illustrates how candidates are evaluated against the criteria.

Verity Spider Command

In this example, the following Verity Spider command will be used.

```
vspider -cmdfile c:\verity\jobs\accounting.cmd
```

where accounting.cmd contains the following:

```
-collection memo.coll
-style c:\verity\servers\common\styles\vgwhttp
-jobpath c:\verity\jobs\1
-start http://web.verity.com/intdocs
-jumps 2
-domain verity.com
-indinclude *acct*
-indexexclude *memo*
-indmimeinclude application/msword
-indmimeexclude text/html
```

Note The -indinclude option is being used because using -include results in zero files indexed. The reason is the starting point does not qualify according to the -include string. Therefore, the -indinclude option must be used instead, as it will allow following from the starting point while only indexing what matches.

Candidates from Starting Point

This example is not using -include. Therefore you can safely assume the starting point is valid and so start evaluating with index.html in the /intdocs directory.

In this directory...	These files are available for indexing...
/intdocs	index.html, which contains links to files in a subdirectory, accounting.
/intdocs/accounting	acctmemo.doc, acctclose.doc, acctmemo2.doc, acct1.htm, acct2.htm. Both HTML files link to acct1.doc and acct2.doc.

Evaluating the Candidates

The URL `http://web.verity.com/intdocs` is processed, and the file `index.html` is retrieved because the Web server is configured to serve `index.html` for the virtual directory `/intdocs`.

Processing `index.html`

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `index.html` fails. At this point, `index.html` will not be included for indexing. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `index.html` does not contain those characters so it passes. However, the initial criteria still holds meaning at this point, `index.html` is still not to be included for indexing. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `index.html` fails this criteria. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `index.html` fails.
5. The conclusion is `index.html` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `index.html` is examined for links to other files and other links.

Following is a simplistic representation of a part of `index.html`, to illustrate the links and documents being evaluated in this example.

```
<body>
<a href="accounting/acctmemo.doc">Memo Month 1</a>
<a href="accounting/acctmemo2.doc">Memo Month 2</a>
<a href="accounting/acctclose.doc">Accounts Closed</a>
<a href="accounting/acct1.htm">First half of month</a>
<a href="accounting/acct2.htm">Second half of month</a>
</body>
```

Each of the files are then processed in turn, one at a time.

Processing acctmemo.doc

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctmemo.doc` passes. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctmemo.doc` fails. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acctmemo.doc` passes this criteria. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctmemo.doc` passes.
5. The conclusion is `acctmemo.doc` cannot be included for indexing. Although it passes three of the criteria, it cannot be indexed because all criteria must be passed.

Processing acctmemo2.doc

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctmemo2.doc` passes. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctmemo2.doc` fails. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `application/msword` are to be included for indexing. `acctmemo2.doc` passes this criteria. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctmemo2.doc` passes.
5. The conclusion is `acctmemo2.doc` cannot be included for indexing. Although it passes three of the criteria, it cannot be indexed because all criteria must be passed.

Processing acctclose.doc

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctclose.doc` passes. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctclose.doc` passes. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `application/msword` are to be included for indexing. `acctclose.doc` passes. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctclose.doc` passes.
5. The conclusion is `acctclose.doc` is to be included for indexing.

Processing acct1.htm

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct1.htm` passes. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct1.htm` passes. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `application/msword` are to be included for indexing. `acct1.htm` fails. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct1.htm` fails.
5. The conclusion is `acct1.htm` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `acct1.htm` is examined for links to other files and other links.

Following is a simplistic representation of a part of `acct1.htm`, to illustrate the links and documents being evaluated in this example.

```
<body>  
<a href="accounting/acct1.doc">Accounts 1st month</a>  
</body>
```

The file `acct1.doc` will be processed after `acct2.htm`.

Processing acct2.htm

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct2.htm` passes. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct2.htm` passes. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `application/msword` are to be included for indexing. `acct2.htm` fails. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct2.htm` fails.
5. The conclusion is `acct2.htm` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `acct1.htm` is examined for links to other files and other links.

Following is a simplistic representation of a part of `acct2.htm`, to illustrate the links and documents being evaluated in this example.

```
<body>
<a href="accounting/acct2.doc">Accounts 2nd month</a>
</body>
```

The file `acct2.doc` is then processed with `acct1.doc`.

Processing `acct1.doc` and `acct2.doc`

1. `-indinclude *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
2. `-indexclude *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
3. `-indmimeinclude application/msword` — This criteria states that only files that are of the MIME Type `application/msword` are to be included for indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
4. `indmimeexclude text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct1.doc` and `acct2.doc` pass.
5. The conclusion is `acct1.doc` and `acct2.doc` are to be included for indexing.

NOTES

The following table summarizes what happened with each file.

Candidate	-indinclude *acct*	-indexclude *memo*	-indmimeinclude application/msword	-indmimeexclude text/html	Conclusion
index.html	Fails	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acctmemo.doc	Passes	Fails	Passes	Passes	Not indexed.
acctclose.doc	Passes	Passes	Passes	Passes	Indexed.
acctmemo2.doc	Passes	Fails	Passes	Passes	Not indexed.
acct1.htm	Passes	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acct2.htm	Passes	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acct1.doc	Passes	Passes	Passes	Passes	Indexed.
acct2.doc	Passes	Passes	Passes	Passes	Indexed.

- The ordering of the criteria across the table matches the ordering of the criteria in the command-file example. Although Verity Spider evaluates each criteria option in turn, one at a time, the order is arbitrary.
- The linear nature of processing these files is a product of documenting them. The files can actually be processed simultaneously depending on the number of connections.
- By using the “ind” criteria, the Verity Spider is able to follow links to the limit of -jumps, while still only indexing those files which meet the desired criteria.
- It is technically not necessary to also use -indmimeexclude. Using -indmimeinclude implies that you are excluding everything else.
- It is necessary to have both -indinclude *acct* and -indexclude *memo* because you do not want to index any files which contain both. In most cases, however, using an include value implies excluding everything else, and vice versa.
- When you find that you are not indexing as many files as you think you should, review your indexing criteria to ensure you are not being too restrictive.

The Use of Last-Modified Date

When web crawling to gather HTML documents for indexing, Verity Spider looks for the date the document was last modified, in the form of a field named Last-Modified. The value of Last-Modified is used to determine if documents should be indexed again.

How Last-Modified is Used

How the last-modified value is used depends on the type of document.

New Documents

For HTML documents that have never been indexed, the value for Last-Modified is not compared to anything as nothing exists in the persistent store. Instead, the Last-Modified date, if it exists, is stored in the `last_modified_date` field of the persistent store for the collection into which the document is being indexed.

Previously Indexed Documents

For HTML documents which have been indexed, and for which a value exists in the `last_modified_date` field in the persistent store, Verity Spider compares the retrieved document's Last-Modified value with `last_modified_date`. What happens to the document depends on the outcome of the comparison.

- If no value exists in the persistent store, or it is older than the current value, then the document is indexed into the collection and the `last_modified_date` field is updated in the persistent store.
- If a value exists in the persistent store and it is the same or newer than the current value provided by the retrieved document, then the document is not indexed.
- If a Last-Modified value is not provided with a retrieved document, then Verity Spider treats the document as new and always indexes it.

Dynamic Documents

If you are dealing with dynamically generated HTML documents, then there may never be a Last-Modified date and so the document may always be indexed. A workaround is to incorporate a meta tag into the processing of the dynamic documents and take advantage of the `-metafile` option.

Verity Spider Examples

This chapter contains some indexing examples with specific criteria, and some in-depth examples covering specific situations.

This chapter includes the following sections:

- [Examples](#)
- [Specific Situations and Concepts](#)

Examples

The first section of this chapter contains examples of running the Verity Spider with various options from the command-line. For more in-depth examples, see [“Specific Situations and Concepts” on page 129](#).

Each of the examples is prefaced with a description of the indexing job requirements. Remember that you can always include options which are not explicitly mentioned in these examples. The options which are provided satisfy the individual case requirements and should be viewed as a framework only.

- [Deleting Linked Documents](#)
- [Skipping Documents](#)
- [Preferring a Site for Duplicates](#)
- [Reparsing a Site](#)
- [Omitting Optimization](#)
- [Indexing Virtual Hosts](#)
- [Updating Only Certain Documents](#)
- [Custom Value for Last-Modified Date](#)
- [An Intranet with CGI](#)
- [Web Sites and Proxy Servers](#)
- [Adding to an Existing Collection](#)
- [Including Previously Dropped Documents](#)
- [File Systems](#)

Deleting Linked Documents

In HTML documents, links to referred documents are sometimes changed or deleted. For example, an HTML document that originally contains links to five documents is updated such that the link to one particular document is deleted. The document itself, however, still resides in the file system.

Because the document itself still exists in the file system, `vspider` does not consider it deleted and maintains knowledge of the document in its persistent store. The document is also still available for searching from the collection into which it was indexed.

Simply removing the link in a referring document is not enough to remove the document from search results. In addition to removing the link, you must also remove the document from the file system (available to the web server serving those files) for `vspider` to consider it deleted and no longer available for searches.

Skipping Documents

You want to crawl an entire web site, where you want to parse for links to other documents but not actually index any HTML document which contains the text “welcome” in the <Title> tag.

```
vspider -cmdfile /verity/spider/skip1.cmd
```

where skip1.cmd consists of:

```
-collection icd.coll  
-start http://www.mysite.com  
-style /verity/mystyles/custom  
-indskip title "welcome"
```

Table 4-1 Case-specific Options

Option	Reason
-indskip	Enables Verity Spider to parse for links yet not actually index documents which meet the specified criteria.

Preferring a Site for Duplicates

You want to crawl a pair of web sites, where you want to specify the geographically closer server for retrieving documents for viewing when duplicate documents are detected on both servers. Duplicate detection is automatically enforced.

```
vspider -cmdfile /verity/spider/prefer.cmd
```

where `prefer.cmd` consists of:

```
-collection icd.coll  
-start http://www.thesite1.com  
-start http://www.thesite2.com  
-style /verity/mystyles/custom  
-preferred thesite1.com
```

Table 4-2 Case-specific Options

Option	Reason
-preferred	Enables Verity Spider to specify a particular server for retrieving documents for viewing when duplicates are detected on multiple servers.

Reparsing a Site

Suppose you index only HTML documents on a site.

```
vspider -cmdfile /verity/spider/allhtml.cmd
```

where `allhtml.cmd` consists of:

```
-collection allhtml.coll  
-style /verity/mystyles/vgwhhttp  
-start http://www.mysite.com  
-mimeinclude text/html
```

Now you want to update the collection with all of the other document types linked to on those HTML pages. You do not need to specify `-style` because you are updating an existing collection which already contains style files.

Note The following command must be issued as a single line from the command-line. It is broken up here for readability.

```
vspider -collection allhtml.coll  
-jobpath /usr/verity/jobs/2  
-start http://www.mysite.com  
-reparse
```

Table 4-3 Case-specific Options

Option	Reason
-reparse	Forces Verity Spider to crawl the HTML documents again, indexing any documents which are allowed.

Table 4-4 Unnecessary Options for this Case

Option	Reason
-mimeinclude	In order for Verity Spider to have anything to do when you use <code>-reparse</code> , you must either omit previous exclusion criteria, or introduce new inclusion criteria. In this case, specifying only HTML in the original job excluded all other file types. By omitting <code>-mimeinclude</code> in the second job with <code>-reparse</code> , you will index all document types to which there are links in the HTML documents.

Omitting Optimization

You do not want the overhead of optimization during an indexing job, because you plan to do your own customizing manually with `mkvdk`.

```
vspider -cmdfile /verity/spider/solo.cmd
```

where `solo.cmd` consists of:

```
-collection icd.coll  
-start http://www.mysite.com  
-style /verity/mystyles/vgwhhttp  
-nooptimize
```

Once the job has completed, you need to run `mkvdk` to optimize the collection. For example, you might run:

```
mkvdk -collection icd.coll -optimize maxclean -vdbopt
```

For more information on `mkvdk`, see the *Verity Collection Reference*.

Table 4-5 Case-specific Options

Option	Reason
-nooptimize	Forces Verity Spider to forego optimization. Keep in mind that you should never leave a collection unoptimized for too long, as search times can begin to suffer.

Indexing Virtual Hosts

You want to index both `www.mysite.com` and `search.mysite.com`, and they are both DNS aliases of `webhost.mysite.com`. Webhost is the canonical name of the physical machine on which the web server hosting the aliases is running. The web server is set up to serve `http://www.mysite.com` and `http://search.mysite.com` with different document roots.

```
vspider -cmdfile /verity/spider/vhosts.cmd
```

where `vhosts.cmd` consists of:

```
-collection icd.coll  
-jobpath /usr/verity/jobs/4  
-start http://www.mysite.com http://search.mysite.com  
-style /verity/mystyles/vgwhhttp  
-virtualhost www.mysite.com search.mysite.com
```

Table 4-6 Case-specific Options

Option	Reason
-virtualhost	You want to index multiple sites running on the same server. Without -virtualhost, only the documents from <code>www.mysite.com</code> would be indexed whenever a duplicate file name also existed for <code>search.mysite.com</code> . This is because a DNS lookup would resolve both sites to Webhost and the documents would be considered duplicate based on name.
-jobpath	You want the job directory to exist somewhere other than the default location, inside the current collection.

Updating Only Certain Documents

You want to update a large collection, but only with those documents that were last indexed at least 30 hours ago. You do not need to specify `-style` because you are updating an existing collection which already contains style files.

```
vspider -cmdfile /verity/spider/update.cmd
```

where `update.cmd` consists of:

```
-collection icd.coll  
-refresh  
-refreshtime 1 day 6 hours
```

Table 4-7 Case-specific Options

Option	Reason
-refreshtime	Since it is a large collection, you do not want to index documents that were indexed in the last 30 hours. Using <code>-refreshtime</code> enables you to specify a time threshold for documents to refresh.
Another way to specify 30 hours is to simply use only hours:	
<pre>-refreshtime 30 hours</pre>	

Custom Value for Last-Modified Date

You want to use a custom meta tag in HTML documents to replace the Last-Modified date value returned by the web server. This example is overly simplistic to highlight the usage of `-metafile`. Your indexing jobs will likely differ.

```
vspider -cmdfile /verity/spider/meta.cmd
```

where `meta.cmd` consists of:

```
-collection icd.coll -metafile /verity/dlt.txt  
options
```

and `dlt.txt` consists of the following:

```
Document_Last_Touched Last-Modified
```

If you want the `Document_Last_Touched` value to always take precedence, add the override flag “Y” at the end of the entry. If you want the `Document_Last_Touched` value to be used only when the web server does not itself provide a value for `Last_Modified`, then add the override flag “N” at the end of the entry.

The custom meta tag `Document_Last_Touched` must exist in all HTML documents, and contain a date value in one of the following date formats:

Date format	Example
RFC822 (updated by RFC 1123)	Example: Sun, 06 Nov 1994 08:49:37 GMT
RFC850 (obsoleted by RFC 1036)	Example: Sunday, 06-Nov-94 08:49:37 GMT
ANSI C's <code>asctime()</code> format	Example: Sun Nov 6 08:49:37 1994
Warning! The day value must occupy two spaces. If you only have one digit, as in the example, then you must provide an extra space between the month and the digit.	

Table 4-8 Case-specific Options

Option	Reason
<code>-metafile</code>	This option specifies the mapping file which contains the information mapping your custom meta field to the standard HTTP header field, Last-Modified.

An Intranet with CGI

You want to index your internal web servers, one UNIX and one Windows, both of which host CGI scripts to dynamically create web pages. You also want complete logging of all messages.

```
vspider -cmdfile /verity/vspider/intra.cmd
```

where `intra.cmd` consists of:

```
-collection icd.coll  
-start http://sigma.verity.com:8015  
-start http://colt.verity.com  
-style /verity/mystyles/vgwhhttp  
-domain verity.com  
-cgiok  
-loglevel trace
```

Table 4-9 Case-specific Options

Option	Reason
-start	The URL contains the host name and domain name. When indexing secure web sites, you must include the domain name.
-cgiok	You must explicitly enable the Verity Spider's ability to parse documents served by CGI scripts.
-domain	You do not want to go beyond the local domain (in this case, <code>verity.com</code>) to gather documents.
-loglevel trace	You must explicitly state a higher logging level than the default <code>verbose</code> , and in this case the most complete level is <code>trace</code> .

Table 4-10 Unnecessary Options for this Case

Option	Reason
-nofollow, -unlimited	You want to index links, but only on those servers which are within your domain. Therefore <code>-nofollow</code> cannot be used to stop Verity Spider from following links, and <code>-unlimited</code> would undermine your desire to only follow links to hosts within your domain.

Web Sites and Proxy Servers

You want to index only static documents on both an internal and an external web site. The external web site must be accessed through a proxy server that requires authentication, and the internal web site can be accessed without a proxy server.

```
vspider -cmdfile /verity/vspider/proxy.cmd
```

where `proxy.cmd` consists of:

```
-collection icd.coll  
-start http://host.verity.com:8015 -start http://www.company.com  
-style /verity/mystyles/vgwhhttp  
-noproxy '*.verity.com'  
-proxy proxyhost:8080  
-proxyauth jcameron:1912sunk  
-timeout 10 -jumps 20 -pathlen 8 -indexers 4 -connections 10
```

Table 4-11 Case-specific Options

Option	Reason
-noproxy	Since you know that the internal site can be accessed without a proxy server, you can optimize the indexing job by explicitly instructing Verity Spider to not attempt to use a proxy server.
-proxy	Since you know that the external site can only be accessed through a proxy server, you explicitly instruct Verity Spider to use the indicated host and port.
-proxyauth	In order to get through the secure proxy server, that requires authentication, specified with <code>-proxy</code> , you must include a username and password with <code>-proxyauth</code> .

Adding to an Existing Collection

You want to add only Microsoft Word and Excel documents to an existing collection.

Note The following command must be issued as a single line from the command-line. It is broken up here for readability. You do not need to specify `-style` because you are updating an existing collection which already contains style files.

```
vspider -collection icd.coll
-start f:\documents
-indmimeinclude application/msword
-indmimeinclude application/excel
```

Table 4-12 Case-specific Options

Option	Reason
-indmimeinclu de	This option specifies that only the specified MIME Types are to be indexed. In this case, an additional instance of <code>-indmimeinclude</code> is necessary to also index a second MIME Type. You could also include all values in a single instance of <code>-indmimeinclude</code> .

Table 4-13 Unnecessary Options for this Case

Option	Reason
-indmimeexclu de	When you use inclusion criteria, it is implied that all other criteria are excluded.
All Networking Options, -cgiok, -norobo	These options only affect indexing web sites.

Including Previously Dropped Documents

You indexed a site during which several hosts timed out and could not provide necessary documents. You have since learned that the hosts in question were undergoing maintenance and are now available.

Note The following command must be issued as a single line from the command-line. It is broken up here for readability. You do not need to specify `-style` because you are updating an existing collection which already contains style files.

```
vspider -collection icd.coll  
-refresh  
-host internal.verity.com -host marketing.verity.com  
-timeout 90  
-delay 30000 -retry 6
```

Table 4-14 Case-specific Options

Option	Reason
-refresh	This option specifies an incremental indexing job.
-host	This option restricts indexing to only those hosts which are specified. In this case , they are the hosts you know were not available when the original job ran. When you use <code>-restart</code> , you can use <code>-start</code> or at least one of <code>-host</code> , <code>-domain</code> , <code>-nofollow</code> , or <code>-unlimited</code> .
-timeout	By increasing the amount of time before a request times out, you increase the chances that Verity Spider will be able to maintain the connection with the host and retrieve documents.
-delay	By specifying a delay in the http requests, you decrease the chances that the hosts will be overwhelmed.
-retry	By specifying <code>-retry</code> , you increase the chances that a document will be retrieved. The default value is 4.

File Systems

You want to spider a network drive to index all Microsoft Word and ASCII text documents, while skipping all directories named TEMP, CONFIDENTIAL, and ACCOUNTING.

```
vspider -cmdfile c:\verity\vspider\files.cmd
```

where files.cmd consists of:

```
-collection icd.coll  
-start f:\documents  
-style C:\verity\mystyles\vgwfsys  
-indmimeinclude 'application/msword' -indmimeinclude 'text/plain'  
-exclude TEM* CONFIDENTI* ACCOUNT*
```

If you find MIME Types are being dropped, or you know you will be indexing files whose extensions are not known to the Verity Spider by default, use the regular expression `'*/*'` for your MIME criteria.

For example:

```
-mimeinclude */*
```

Furthermore, you should also use inclusion and exclusion criteria to fine tune what is indexed.

- If your list of file types to index is rather long, use one of the exclusion criteria (`-exclude`, `-indexexclude`, `-mimeexclude`, or `-indmimeexclude`) to exclude extensions you know you do not want to index. For example:

```
-exclude *.exe *.com
```

- If the list of file types you want to index is relatively small, use one of the inclusion criteria (`-include`, `-indinclude`, `-mimeinclude`, or `-indmimeinclude`) to specify them. For example:

```
-indinclude *.txt *.lst *.log
```

Table 4-15 Case-specific Options

Option	Reason
-indmimeinclude	This option specifies that only the specified MIME Types are to be indexed. Although other files may be gathered, such as HTML pages which contain links to the desired document types, only those types which are indicated are actually indexed. In this case, an additional instance of -indmimeinclude is necessary to also index a second MIME Type. You could also include all values in a single instance of -indmimeinclude.
-exclude	This option enables you to control what is followed for gathering.

Table 4-16 Unnecessary Options for this Case

Option	Reason
-indmimeexclude	By default, the Verity Spider will only index those document types specified with indmimeinclude, automatically ignoring all others. Therefore it is not necessary to explicitly specify that the text/html MIME Type be excluded.
All Networking Options, -cgiok, -norobo	These options only affect indexing web sites.

Specific Situations and Concepts

The examples in this section cover specific situations and concepts.

- [Customizing the Last-Modified Date](#)
- [Indexing Secure Web Sites](#)
- [Indexing with Proxy Servers](#)
- [Indexing Network, UNC and Secure Paths on Windows](#)
- [Defining the Authenticating Repository for Secure File System Collections](#)
- [Access Control List \(ACL\) Caching — Domain versus Local Users](#)
- [Prefix Mapping](#)
- [Setting MIME Types](#)
- [Using StyleSet Editor in Standalone Mode](#)

Customizing the Last-Modified Date

There may be cases in which you want to customize the Last-Modified date value.

Scenarios for Customizing

There are two scenarios under which you would want to customize the Last-Modified value. These are:

- The Last-Modified meta tag is missing, requiring you to add one.

There are two possible reasons for the Last-Modified meta tag to be missing. These are:

- The documents are created dynamically (on-the-fly).
- The web server is configured to omit the value.

Under this scenario, you will not be able to use the Last-Modified meta tag, or any variation thereof. In order to create a custom value, or override an existing value, the Last-Modified meta tag must exist in the HTML document itself as it is retrieved from the web server. If you do not have any control over the web server or documents in question, you can still exercise some control over when documents are indexed by using `-refreshtime` and `-refresh` with inclusion or exclusion criteria. For more information on these options, see [“Reference of Command-Line Options” on page 38](#).

- The value provided does not contain a satisfactory value, and you want to override it. You can do this for file system indexing as well as web site indexing.

Whether you need to insert a value for Last-Modified, or you want to override an existing value, the steps are the same.

Example for Customizing a Value

You want to use a custom meta tag in HTML documents to replace the Last-Modified date value returned by the web server. This example is overly simplistic to highlight the usage of `-metafile`. Your indexing jobs will likely differ.

```
vspider -cmdfile /verity/spider/meta.cmd
```

where `meta.cmd` consists of:

```
-collection icd.coll -metafile /verity/dlt.txt  
options
```

and `dlt.txt` consists of the following:

Document_Last_Touched Last-Modified

If you want the Document_Last_Touched value to always take precedence, add the override flag “Y” at the end of the entry. If you want the Document_Last_Touched value to be used only when the web server does not itself provide a value for Last_Modified, then add the override flag “N” at the end of the entry.

The custom meta tag Document_Last_Touched must exist in all HTML documents, and contain a date value in one of the following date formats:

Date format	Example
RFC822 (updated by RFC 1123)	Example: Sun, 06 Nov 1994 08:49:37 GMT
RFC850 (obsoleted by RFC 1036)	Example: Sunday, 06-Nov-94 08:49:37 GMT
ANSI C’s asctime() format	Example: Sun Nov 6 08:49:37 1994
Warning! The day value must occupy two spaces.If you only have one digit, as in the example, then you must provide an extra space between the month and the digit.	

Table 4-17 Case-specific Options

Option	Reason
-metafile	This option specifies the mapping file which contains the information mapping your custom meta field to the standard HTTP header field, Last-Modified.

How to Customize the Last-Modified Date Value

Review the steps below for a description of how to customize the Last-Modified date value.

Step 1 — Add a Meta Tag

The first step is to ensure that your HTML documents contain a meta tag which can be used for the Last-Modified date. If your documents do not contain such a meta tag, you will need to add one.

Meta tags use the syntax:

```
<meta name="name" content="content">
```

where *name* is any string which you will later use in the text map file, and *content*, is the date and time you want to use for Last-Modified. The value for *content* must be in one of the following date formats:

Date format	Example
RFC822 (updated by RFC 1123)	Example: Sun, 06 Nov 1999 08:49:37 GMT
RFC850 (obsoleted by RFC 1036)	Example: Sunday, 06-Nov-99 08:49:37 GMT
ANSI C's asctime() format	Example: Sun Nov 6 08:49:37 1999
Warning! The day value must occupy two spaces.If you only have one digit then you must provide an extra space between the month and the digit.	

For dynamic documents, such as from database middleware or scripts, you will have to find a way to incorporate the meta tag into the processing of the document.

Step 2 — Create a Text File

Since Verity Spider by default only recognizes the valid HTTP header Last-Modified along with its value, you will need to map your custom meta tag name to the expected name Last-Modified.

In a text editor, create a simple text file with a single entry for mapping your custom meta tag name to the expected name, Last-Modified. The syntax for the entry is:

```
name Last-Modified Y/N
```

where *name* is the meta tag name you specified in step 1, and Y/*N* is an override flag which can be either “Y” for yes or “N” for no.

Flag	Description
Y	When you use the “Y” flag, the value for the custom meta tag overrides the value for Last-Modified, even if both values are present and differ.
N	When you use the “N” flag, the value for the custom meta tag will be used only if there is no value for Last-Modified. If a value for Last-Modified exists, then that is given precedence.

Step 3 — Run a Verity Spider indexing job with the `-metafile` option

Once you have your custom meta tag name in the HTML documents and you have created a text file in which you map this custom meta tag name to the expected meta tag name Last-Modified, you can run a Verity Spider indexing task using the `-metafile` option.

For example:

```
vspider -collection collA -style path/mystyles  
-metafile path/mapmeta.txt
```

Remember that the simple text file `mapmeta.txt` contains the mapping for your custom meta tag to the Last-Modified meta tag.

Indexing Secure Web Sites

Some sites secure documents by requiring authentication on the realm in which they are stored. This is typically accomplished using the administration functions of the Web server. In order to retrieve documents for indexing from such secured directories, the Verity Spider can use the `-auth` option to reference an authorization file. The authorization file is a simple ASCII text file that you must create.

Note Verity Spider automatically supports Basic, NTLM and Public authentication with Microsoft's Internet Information Server Web server. These authentication mechanisms are also automatically supported when Verity needs to retrieve the documents for viewing.

Procedure

Indexing web sites secured by realms involves the following tasks:

1. Prepare an authorization file for later use with the `-auth` option.

The authorization file is an ASCII text file that you must create manually in a text editor. The authorization file contains one record per line. Each line consists of the following columns, separated by white space.

- Server name, including port number
- URL path
- User name
- Password

Here is a sample authorization file:

#Server Name	URL Path	Username	Password
cairo.myco.com:8080	/docs	Aladdin	open
www.123456.com	/arts/columns	barelto	94089

For the sake of simplicity, you can save the authorization file in the `/style` directory of the collection into which you are indexing, and use a name such as `vgwhttp.auth`.

Note If you have a large number of web sites that share the same credentials, you can specify the following in your authorization file:

#Server Name	URL Path	Username	Password
*	" "	Aladdin	open

This will allow Verity Spider to use the credentials `Aladdin` and `open` for all web sites specified in the indexing job for which you are using this authorization file.

2. Create a `vgwhttp.cfg` file in which you define the secured realms so documents can be retrieved based on results list filtering and for viewing.

For more information, see [“The vgwhttp.cfg File” on page 136](#).

3. Run an indexing job and specify the `-auth` option, with the appropriate authorization file.

For example:

```
vspider -collection C:\Colls\newcoll -style C:\styles\webset1  
-auth C:\styles\webset1\style\vgwhttp.auth  
-start http://host1.domain.com  
otheroptions
```

Note You must include a domain name in the URL for your starting points.

4. Search the collection and view the documents retrieved from the web sites.

The vgwhttp.cfg File

When you specify the `-auth` option for an indexing job, you are only configuring the ability for Verity Spider to gather documents to be indexed into a collection. To support results list filtering and viewing, you must specify the appropriate information in an HTTP gateway configuration file, called `vgwhttp.cfg`.

Note There may already be a `vgwhttp.cfg` file in your style set. These instructions assume you are creating one from scratch. If one already exists, just copy the relevant information described in these instructions into the appropriate section of the existing `vgwhttp.cfg` file.

1. In a text editor, open a new, blank document and type the following information:

```
# vgwhttp.cfg - HTTP gateway configuration file  
$control:1
```

Note If you are editing an existing `vgwhttp.cfg` file, due to adding more starting points to an existing collection for example, you can proceed to step 3 and just add more `repository` sections as necessary.

If you are also using `-proxy`, `-proxyauth` or `-noproxy` in your indexing job, you must also specify information relevant to those options in the `vgwhttp.cfg` file. For more information, see [“Indexing with Proxy Servers” on page 140](#).

You would add the proxy-related information after the closing curly bracket `}` of the last `repository` section, and before the end-of-file designation of `$$`.

2. Save the text file as `vgwhttp.cfg` in the `/style` directory for the collection associated with the indexing job in which you specified the `-auth` option.
3. In your `vgwhttp.cfg` file, add a `repository` section for each group of URLs for which the same credentials apply.

For example, in your authorization file, you might have the same credentials (`user1`, `password1`) for several different realms. You can group the URLs that correspond to those realms into a single `repository` section.

repository Section Syntax

The syntax for a `repository` section is as follows. You can specify multiple `repository` sections.

```
repository: name
{
    securityModuleId: mod_id
    url: protocol://host:port/path/*.
    loginurl: hostnameonly:port username password
}
```

name — The repository name is an arbitrary label used to identify the URL or group of URLs that share a set of credentials. Use any unique alphanumeric string that enables you to easily identify this `repository` section.

mod_id — The *mod_id* defines whether the security module is unsecured or secured. Valid values are:

`0x0` — Use this value if you do not want results filtered based on user. In other words, you would allow a user to see all possible results, while still restricting document access at view time.

`0xdff4` — Use this value if you want to filter results so that a user cannot even see results unless he or she has the proper access.

protocol://host:port/path/.* — Specify the *protocol* (`http` or `https`), host name (*host*), port number (*port*), and path (*path*) that correspond to the realm that requires authentication.

4 Verity Spider Examples

Specific Situations and Concepts

For example, a web server may define the realm `NeedToKnow` as the path:

```
http://cairo.myco.com:8000/restricted/docs
```

You would then specify the following for `protocol://host:port/path/.*`:

```
http://cairo.myco.com:8000/restricted/docs/.*
```

`loginurl: protocol://host:port/path`— The `loginurl` entry enables you to define a path against which you can test a user's credentials. If the credentials fail on the value for `loginurl`, then the gateway will prompt the user for the proper credentials if the search application enables it. Otherwise, nor results are returned to that user.

Note You can only specify the specific URL path. Do not include any wildcard expressions.

For example, if you have several realm URLs for the server `cairo.myco.com:8000`, such as

```
http://cairo.myco.com:8000/data/fyio
```

and

```
http://cairo.myco.com:8000/otherdata/fyio
```

you can define a `loginurl` entry as:

```
loginurl: http://cairo.myco.com:8000
```

This feature helps to improve the performance of the gateway when results list filtering is implemented.

Sample vgwhhttp.cfg File

Following is a sample `vgwhhttp.cfg` file based on this information:

- Realm 1 is defined as: `http://cairo.myco.com:8000/data/fyio`
- Realm 2 is defined as: `http://cairo2.myco.com:8008/otherdata/fyio`
- You want to filter results based on credentials

Here is the `vgwhhttp.cfg` file:

```
# vgwhhttp.cfg - HTTP gateway configuration file
$control:1
repository: data
{
    securityModuleId: 0xdff4
    url:http://cairo.myco.com:8000/data/fyio/*.
    loginurl: http://cairo.myco.com:8000/data/fyio
}
repository: otherdata
{
    securityModuleId: 0xdff4
    url:http://cairo2.myco.com:8008/otherdata/fyio/*.
    loginurl: http://cairo2.myco.com:8008/otherdata/fyio
}
$$
```

Indexing with Proxy Servers

Some web sites cannot be accessed without going through a proxy server, which may also be secured and require authentication. In order to retrieve documents for indexing from such web sites, the Verity Spider can use the `-proxy` and `-proxyauth` options.

If there are any web sites that can be accessed directly, without having to go through the proxy server, then you can use the `-noproxy` option to specify them and include them in the same indexing job.

Procedure

Indexing web sites secured behind a proxy server involves the following tasks:

1. Create a `vgwhttp.cfg` file.

In the `vgwhttp.cfg` file, you will specify the information you will later use with the `vspider` options `-proxy`, `-proxyauth` and `-noproxy` so that documents can be retrieved for viewing.

For more information, see [“The vgwhttp.cfg File” on page 141](#).

2. Run an indexing job and specify the `-proxy`, `-proxyauth` and `-noproxy` options.

For example:

```
vspider -collection C:\Colls\newcoll -style C:\styles\webset1  
-proxy proxyhost:port -proxyauth username:password -noproxy  
host_1 [host_n]  
otheroptions
```

3. Search the collection and view the documents retrieved from the web sites.

The vgwhhttp.cfg File

When you specify the `-proxy`, `-proxyauth` and `-noproxy` options for an indexing job, you are only configuring the ability for Verity Spider to gather documents to be indexed into a collection. To support results list viewing, you must specify the appropriate information in an HTTP gateway configuration file, called `vgwhhttp.cfg`.

Note There may already be a `vgwhhttp.cfg` file in your style set. These instructions assume you are creating one from scratch. If one already exists, just copy the relevant information described in these instructions into the appropriate section of the existing `vgwhhttp.cfg` file.

1. In a text editor, open a new, blank document and type the following information:

```
# vgwhhttp.cfg - HTTP gateway configuration file
$control:1
```

Note If you are editing an existing `vgwhhttp.cfg` file, due to adding more starting points to an existing collection for example, you can proceed to step 3 and just add more `repository` sections as necessary.

If you are also using `-auth` in your indexing job, you must also specify information relevant to that option in the `vgwhhttp.cfg` file. For more information, see [“Indexing Secure Web Sites” on page 134](#).

You would add the authorization-related information after the closing curly bracket `}` of the proxy section, and before the end-of-file designation of `$$`.

2. Save the text file as `vgwhhttp.cfg` in the `/style` directory for the collection associated with the indexing job in which you will be specifying the `-proxy`, `-proxyauth` and `-noproxy` options.
3. In your `vgwhhttp.cfg` file, add a proxy section with the relevant information.

The syntax for a proxy section is described in the next section,

proxy Section Syntax

The syntax for a proxy section is as follows:

```
proxy: hostname portnum
{
  proxyAuth: username password
  noproxy:
  {
    server: hostname_or_IP_address
  }
}
```

hostname portnum — The host name and port number of the proxy server.

username password — The credentials required for a secured proxy server.

This information is optional. Leave the `proxyAuth` line out if your proxy server does not require credentials.

hostname_or_IP_address — The host name or IP address of web servers that can be accessed directly from the computer on which `vspider` is running without having to go through the specified proxy server.

Note Do not specify a port number; only the host name or IP address of a web server.

You can specify up to 255 entries for `server:hostname_or_IP_address`. You can use the question mark (?) and asterisk (*) wildcard characters to define the *hostname_or_IP_address*, where ? represents a single character, and * represents a string of characters.

The `noproxy` entry and the `server` sub-entry are optional. If you do not need to define any directly accessible hosts, you can omit the entries

```
noproxy:
{
  server: hostname_or_IP_address
}
```

Sample vgwhttp.cfg File

Following is a sample `vgwhttp.cfg` file based on this information:

- You have a proxy server `caliber.myco.com` at port 9000
- The proxy server `caliber.myco.com` requires the credentials `admin1 psswd`
- You can directly access the web servers `quinn.myco.com` and `carly.myco.com`

Here is the `vgwhttp.cfg` file:

```
# vgwhttp.cfg - HTTP gateway configuration file
$control:1
proxy: caliber.myco.com 9000
{
    proxyAuth: admin1 psswd
    noproxy:
    {
        server: quinn.myco.com
        server: carly.myco.com
    }
}
$$
```

Indexing Network, UNC and Secure Paths on Windows

In order to index mapped network drives, UNC paths and secure file system paths with the File System Gateway on Windows, you must ensure that you specified a user account and not the local system account for the Administration Server during installation and you must run `vspider.exe` as the same user with the rights described below.

For more information, see the *Verity K2 Installation and Setup Guide*.

Note When you index network mounted drives on Windows, you should use UNC paths (`\\host\path`) whenever possible as they provide a more reliable access method than mapped drive letters that may change after re-boots.

Running `vspider.exe`

You must run `vspider.exe` as a user with the following rights:

- act as part of the operating system
- increase quotas
- log on as a service
- replace process level tokens
- bypass traverse checking (this is set by default)
- create a token object

The user must be a local user, or a Domain user in a trusted Domain with local Administrator group privileges, and have the necessary access rights to the paths to be indexed.

Note Make sure the user has these rights and not the group the user belongs to. Occasionally, giving groups the rights is not sufficient. Also, you should reboot your computer if you had to add these rights.

With `vspider` running in the proper user context, you will be able to access mapped network drives and UNC paths as starting points to index.

Defining the Authenticating Repository for Secure File System Collections

For secure file system collections, you must have a repository, or host name, that authenticates users attempting to search for and view documents in the collections. This authenticating repository is defined in the configuration file, `vgwfsys.cfg`.

By default, the repository name used to authenticate is the local host name. If you replicate a secure file system collection to other hosts, you should specify an independent authenticating repository.

To specify an independent authenticating repository, edit the `vgwfsys.cfg` file by uncommenting the `repositoryName:` line and adding the appropriate host name. For example:

```
#repositoryName needs to be specified if it needs to be machine
#or host independent
repositoryName: myname
```

Access Control List (ACL) Caching — Domain versus Local Users

By default, ACL caching is enabled for the File System gateway. This means file access information is cached and search users are authenticated against the cached information. When you secure files by local groups only in a Domain environment, ACL caching creates problems for authenticating Domain users.

To work around this problem, you can do one of two things:

- [Disable ACL caching for the File System gateway](#)
- [Secure Files with Domain Users and Groups](#)

Disable ACL caching for the File System gateway

Warning! Disabling ACL caching will result in slower performance during searches because the security check must be performed against the server for every file.

To disable ACL caching, follow these steps:

1. Make a copy of the `fssec` style set located in:

```
installDir/k2/common/styles/fssec
```

where *installDir* is the directory in which you installed K2 Spider

2. In the `vgwfsys.prm` style file in your copy of the `fssec` style set, add an octothorpe (#) as shown to the following line:

```
# $define VGWFSYS_SECURITY_ACLCACHE 1
```

3. Make sure the Domain users who need access to the collection have been added to a local group.

You are now ready to index the collection.

Secure Files with Domain Users and Groups

Use Domain users and groups to secure files, rather than local users or groups. This will allow Domain users to authenticate without problems.

Prefix Mapping

With prefix mapping, you can convert one field, such as a file system path, to another field, such as a URL.

Note When mapping file system paths to URLs, the URLs must also be specified as aliases in your Web server.

Verity Spider Indexing Options

Option	Description
<code>-prefixmap <i>file</i></code>	Specifies a control file which contains source field to destination field mapping information. See “Using the Control File” on page 148 for more information on the content of this file.
<code>-abspath</code>	Forces the Verity Spider to use the absolute path to file system documents when indexing. This option is necessary when mapping file system paths to URLs. It generates document paths that can be understood by a Web server, which would otherwise try to reconcile Verity Spider’s generated relative document paths using the Web server’s document root path as the starting point.

Using the Control File

The control file, which is specified by `-prefixmap`, is a text file which contains the necessary information for mapping a source field to a destination field. A control file consists of the following columns:

Item	Description
SourceField	Field from which values for SourcePrefix will be read.
SourcePrefix	Specifies a prefix, such as a file system path, indexed by the Verity Spider. If the SourcePrefix is a path that includes a trailing slash, then DestPrefix must also include a trailing slash. If the path includes spaces, then enclose it in double quotes. For example: SourcePrefix "C:\My Documents\Files"
DestField	Field where the value modified by DestPrefix will be stored.
DestPrefix	Specifies a destination prefix, such as a Web server alias, used by K2 Server. If the DestPrefix is an alias, it must be created manually in the Web server, before users are allowed to view documents.
Flag	Maps backslashes to forward slashes when specified as a slash <code>"/"</code> . This can be useful with indexed Windows file systems, where a SourcePrefix path ends with a trailing backslash and there are subdirectories.

Prefix Mapping Example 1

In this example, the control file `ctrlhtml.txt` maps a user's Web documents, indexed by way of file system, to a Web server alias for the user. It is assumed that `/search/html` is aliased to `/~search`, with `web` as the Web server document root, for the Web server. The command to run the Verity Spider would look like the following:

```
% vspider -collection /users/colls/pbhtml -start  
          /search/html -abspath -prefixmap  
          /verity/ctrlhtml.txt
```

Each record in the control file must be on a single line. The contents of `ctrlhtml.txt` would look like:

```
# For each document, all lines are considered in order.  
#  
# SourceField  SourcePrefix  DestField  DestPrefix  
VdkVgwKey     /search/html  URL        http://web/~search
```

Prefix Mapping Example 2

In this example, the control file `ctrldoc.txt` maps Microsoft Word documents, indexed by way of file system, to a Web server alias. It is assumed that `d:\docs\isrel` is aliased to `/reldocs`, with `web/` as the Web server document root, for the Web server. Furthermore, the Web server must be configured with a mime-type for Microsoft Word documents. The command to run the Verity Spider would look like the following:

```
c:\>vspider -collection d:\colls\docs -start
          d:\docs\isrel\ -abspath -prefixmap
          d:\verity\ctrldoc.txt
```

Each record in the control file must be on a single line. The contents of `ctrldoc.txt` would look like:

```
# For each document, all lines are considered in order.
#
#SourceField  SourcePrefix    DestField  DestPrefix          Flag
VdkVgwKey    d:\docs\isrel\  URL       http://web/reldocs/  /
```

Since `SourcePrefix` contains a trailing slash, `DestPrefix` does, too. Also, `Flag` is specified here because there may be subdirectories beneath `isrel` which need to have backslashes translated to forward slashes to be served properly by the Web server.

Prefix Mapping Example 3

In this example, PowerPoint slides have been converted to HTML. The conversion to HTML created two files for each slide; a text only `tsld*.htm`, and a text and graphics `sld*.htm`, where “*” is an incrementing number. We want to index the text only files, and then use the text and graphics files for links in a view template.

The control file, `ppview.txt`, will substitute a document file path, and then map the new path to a Web server alias. It is assumed that `c:\dev\slides` is aliased to `/~dev/slides`, with `web/` as the Web server document root, for the Web server. The command to run the Verity Spider would look like the following:

```
c:\>vspider -collection d:\colls\docs -start
          c:\dev\slides -abspath -prefixmap
          d:\verity\ppview.txt
```

Each record in the control file must be on a single line. The contents of `ppview.txt` would look like:

```
# For each document, all lines are considered in order.
#
#
SourceField  SourcePrefix  DestField  DestPrefix  Flag
VdkVgwKey   \dev\slides\t  URL       /dev/slides/  /
VdkVgwKey   c:\dev\slides\ URL       http://web/~dev/slides/ /
```

The first record in `ppview.txt` performs the path substitution required to have the text and graphics files, `sld*.htm`, used as links in a view template instead of the text only files, `tsld*.htm`. The rule of equal trailing slashes for `SourcePrefix` and `DestPrefix` can be disobeyed here because the `DestPrefix` is not a Web URL. The Web URL is mapped in the second record.

With `ppview.txt`, using `URL` in a view template will correctly retrieve the text and graphics document from `c:\dev\slides\` by using the Web server alias `http://web/~dev/slides/`.

Setting MIME Types

You can use the MIME Type criteria options `-mimeinclude`, `-indmimeinclude`, `-mimeexclude` and `-indmimeexclude` to include or exclude MIME Types.

- [Default MIME Types](#)
- [MIME Types Mapped by File Extension](#)
- [Complete List of Known MIME Types](#)
- [Indexing Unknown MIME Types](#)
- [MIME Types and Web Crawling](#)
- [MIME Types and File System Indexing](#)
- [Syntax Restrictions](#)

Default MIME Types

By default, there is a specific list of MIME Types recognized by Verity Spider. When indexing file systems, a file's extension determines the MIME Type. When indexing web sites, a file's extension can determine the MIME Type, but you can configure the relationship as well as add your own using the web server's administration utilities.

MIME Types Mapped by File Extension

When indexing file systems and web sites, the file extensions in the following table are automatically recognized by Verity Spider and are mapped to the corresponding MIME Types. There may be several possible MIME Types for a particular document format. The table below lists the MIME Types with known extensions mapped by Verity Spider.

For these extensions...	Verity Spider maps this MIME Type...
htm, html, shtml, asp, cgi, php, sml	text/html
txt, text, c, h, cpp, cxx, pl, eml	text/plain
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
pdf	application/pdf
mif	application/vnd.mif
rtf	application/rtf
wpd	application/wordperfect5.1
aw	application/applixware
zip	application/zip
mbx	text/x-mbox

Complete List of Known MIME Types

Following is a list of all MIME Types known by Verity Spider by default. Where the previous table listed MIME Types known by file extensions, this is the complete list of MIME Types. Some correspond to the same file extension, while others do not have any known, default file extension.

- application/wita
- application/dec-dx
- application/dca-rft
- application/x-mif
- application/vnd.mif
- application/rtf
- application/vnd.ms-works
- application/macwriteii
- application/wordperfect5.1
- application/vnd.ms-excel
- application/x-excel
- application/excel
- application/x-msexcel
- application/vnd.ms-powerpoint
- application/x-powerpoint
- application/power-point
- application/x-mspowerpoint
- application/msword
- application/applixware
- application/pdf
- application/zip
- text/x-mbox
- text/html
- text/xml
- text/plain

Indexing Unknown MIME Types

Whenever you find MIME Types being dropped, or you know you will be indexing files whose extensions are not known to the Verity Spider by default, you can use the `-mimemap` option or criteria to tune what is to be indexed.

Using `-mimemap`

use the `-mimemap` option to point to a file which contains your own custom mappings for file extensions and MIME Types.

The format for the control file used by `-mimemap` is:

```
#file_ext_no_dot          mime-type
```

To map the unrecognized file extension `abc` to the MIME Type `application/msword`, you would do the following:

1. Create a text file, for example `mappings.txt`, that contains the following line.

```
abc                      application/msword
```

2. Include the `-mimemap` option in your `vspider` command, and refer to the text file that contains the appropriate information from step 1. For example:

```
vspider -mimemap C:\mappings.txt options
```

For more information, see [“Prefix Mapping” on page 147](#).

Using Criteria

You can use inclusion and exclusion criteria to finely control what is indexed.

- If your list of file types to index is rather long, use one of the exclusion criteria (`-exclude`, `-indexexclude`, `-mimeexclude`, or `-indmimeexclude`) to exclude extensions you know you do not want to index. For example:

```
-exclude '*.pdf' '*.pad'
```

- If the list of file types you want to index is relatively small, use one of the inclusion criteria (`-include`, `-indinclude`, `-mimeinclude`, or `-indmimeinclude`) to specify them. For example:

```
-mimeinclude '*/*.txt' '*/*.lst' '*/*.log'
```

MIME Types and Web Crawling

When you index a web site, the Verity Spider evaluates your MIME Type criteria against the "Content-Type" HTTP headers sent by the web server hosting that web site. That web server passes along MIME Type information based on its own internal tables.

When you encounter MIME Types being dropped, make sure the web server you are indexing has the necessary MIME Type information. See the documentation for your web server for information about specifying MIME Types.

You can examine the indexing job's log files for indications that files are being skipped due to MIME Types. For example, a typical ASCII file you might want indexed is a log file (`filename.log`). Unless the web server understands that files with `.LOG` extensions are ASCII text, of MIME Type `text/plain`, you will see in the indexing job log file that `.LOG` files are skipped because of MIME Type even if you use:

```
-mimeinclude "text/*"
```

MIME Types and File System Indexing

When you index a file system, the Verity Spider reads file names and evaluates your MIME Type criteria against an internal, compiled list of known MIME Types and associated file extensions. You can see the complete list in [“Default MIME Types” on page 152](#). You cannot edit this list.

In order to index any MIME Types that are not in the hard-coded, internal list, you must use a combination of the `-mimemap` option and a MIME Type criteria option. For example, you could use `-mimeinclude '*/*' or -indmimeinclude '*/*' to ensure you index all MIME Types. You must also create a text file that maps any non-default MIME Types you want to index, and refer to that text file with the -mimemap option.`

When you encounter MIME Types being dropped, check if the Verity Spider recognizes that particular MIME Type. See the table, [“MIME Types Mapped by File Extension” on page 153](#).

You can examine the indexing job’s log files for indications that files are being skipped due to MIME Types. For example, a typical ASCII file you might want indexed is a log file (`filename.log`). Since the Verity Spider does not understand that files with `.LOG` extensions are ASCII text, of MIME Type `text/plain`, you will see in the indexing job log file that `.LOG` files are skipped because of MIME Type even if you use:

```
-mimeinclude "text/*"
```

Syntax Restrictions

When you specify MIME type criteria, keep in mind the following restrictions.

Using the Wildcard Character (*)

The asterisk (*) wildcard character does not operate as a regular expression for the value of the MIME type criteria. Instead you can only use it to replace the entire MIME type or MIME sub-type.

For example, the following value is a valid substitute for `text/html`:

```
text/*
```

The following value is NOT a valid substitute for `text/html`:

```
text/h*
```

Multiple Parameter Values

When you specify a series of parameter values for a single instance of one of the MIME Type criteria, and you use quotes, you must enclose each separate parameter value in single quotes. For example:

```
-mimeinclude 'text/plain' 'application/*'
```

If you enclose the entire sequence of parameter values,

```
-mimeinclude 'text/plain application/*'
```

the Verity Spider will consider the entire expression as a single value.

You can also use multiple instances of the MIME type criteria, each with a single parameter value, where quotes are necessary only if you use the wildcard character (*).

For example: `-mimeinclude text/plain -mimeinclude 'application/*'`

Using StyleSet Editor in Standalone Mode

You can use the StyleSet Editor in standalone mode, independent of the K2 Dashboard, to create and update your style sets for the Lotus Notes Gateway, Documentum Gateway and Exchange Gateway.

For more information on using the StyleSet Editor in standalone mode, see the respective gateway guide:

- *Verity K2 Notes Gateway Guide*
- *Verity K2 Documentum Gateway Guide*
- *Verity K2 Exchange Gateway Guide*

PART III

K2 Spider

- Chapter 5: K2 Spider Overview
- Chapter 6: Building a K2 Spider System
- Chapter 7: Running K2 Spider Server
- Chapter 8: Using K2 Spider Client
- Chapter 9: Job Initialization File Reference
- Chapter 10: K2 Spider Examples

K2 Spider Overview

This chapter describes the basics about the Verity K2 Spider. The K2 Spider supports multiple jobs and multiple collections per job, includes a job scheduler, and enables the on-line submission of keys, documents, and bulk insert files (BIF) for indexing.

This chapter includes the following sections:

- [K2 Spider Features](#)
- [K2 Spider System](#)

K2 Spider Features

K2 Spider is a client-server model indexer, where you control the process for and structure of gathering data and building collections for use with Verity products. The K2 Spider provides flexibility in configuring your system. It is also scalable to meet your changing requirements.

Overview of Features

K2 Spider offers the following features.

- Distributed gathering and indexing with high scalability and configuration

K2 Spider can scale as your enterprise information demands increase. As your enterprise information grows and propagates, K2 Spider can also scale to support multiple machines and multiple repositories, while ensuring seamless data access within your organization.

The component nature of the K2 Spider Server enables you to run only those components that you want in any particular process space used by the K2 Spider Server (`k2spider_srv`) service. The `k2spider_srv` components are:

- Controller

The Controller provides a single contact point for the K2 Spider Client and is responsible for managing Crawlers, Indexers, indexing jobs, and the workflow of all messages, requests, and data. The Controller cannot share process space with Crawlers or Indexers.

- Crawler

The Crawler fetches document locations and, when you are indexing Web sites, actually downloading the documents themselves.

- Indexer

The Indexer downloads documents, except when you are indexing Web sites, and performs the work of actually indexing information about the documents into a Verity collection. Although each Indexer can only have one collection, you can have as many Indexers as you have collections to index.

- **Flexibility**

K2 Spider provides a superior level of flexibility by allowing you to configure it to run in a single process or multiple processes, on one or many machines. The flexibility of K2 Spider lets you choose how to arrange gathering and indexing jobs. K2 Spider is flexible enough to allow for small, lightweight amounts of data and information, while also being flexible to meet the ever-growing data needs of your business.

- **Comprehensive asset discovery**

K2 Spider locates enterprise assets scattered throughout your organization. Most large enterprise customers have data and documents contained within several file systems, databases, legacy applications, and other repositories that may be located in obscure regions throughout their organizations. K2 Spider enables you to manage this data by locating documents in all types of repositories and making all of your enterprise information accessible to your entire corporate audience.

- **Knowledge repository construction**

K2 Spider enables you to build a knowledge repository with small to infinitely large amounts of data, and efficiently utilize the information required at business critical junctions. The knowledge repositories built with K2 Spider provide simple, effective access to crucial enterprise information.

- **Control of the data indexed**

K2 Spider enables complete control over what data, documents, and information are indexed, along with when the indexing is performed. You also control the types of data to index, frequency of re-indexing, location of where to route specific types of data to be indexed, and other configuration parameters.

- **Supports Secure Socket Layer (SSL) indexing**

In addition to providing access to knowledge contained within HTTP connections, K2 Spider also provides you with the ability to access data contained within secure HTTPS Web sites. K2 Spider also provides full support for 40- and 128-bit encryption and client certificates.

Note This feature requires the purchase of the separate Verity SSL Option Pack. See your Verity sales representative for more information.

- Easy administration

K2 Spider provides a simple, command-line client that makes it easy to administer the gathering and indexing services. The K2 Spider also provides the ability to administer, monitor, and report on gathering and indexing jobs. In addition, K2 Spider provides a set of published APIs that allow the end user to build custom administration and applications components.

- Fault tolerance

While scheduled jobs are running, the Controller maintains a status history on all gathering and indexing services. In the unlikely event that a Crawler or Indexer goes down, the other K2 Spider servers continue operating. The downed server can be re-started seamlessly without interfering with other servers. In the extremely unlikely event that a Controller goes down, you can re-start it without shutting down and re-starting other servers. In this manner, K2 Spider provides fault tolerance for your data collections.

- Flexible document router

K2 Spider lets you route documents to different Indexers based upon your individual requirements. You have a choice of criteria that determine which Indexer will receive your documents.

- Online submission

K2 Spider lets you submit individual documents, document keys or Verity bulk insert files (BIFs) for indexing without re-scheduling, modifying your current jobs, or both. This feature enables you to index data even when the K2 Spider servers lack direct access to it.

K2 Spider System

A K2 Spider indexing system consists of all the separate services and components necessary to create, run, and modify indexing jobs and collections. These services include the K2 Spider Client and the K2 Spider Server. The K2 Spider Server (`k2spider_srv`) is a service which runs as a Controller or in some combination of Crawler and Indexer.

The K2 Spider system is flexible and robust enough to provide a number of combinations for your indexing needs. For example, you can distribute the services and components across multiple machines or run them all on one.

Component Limitations

It is recommended that you run the Controller component alone and not in combination with either of the other components. You may run the Crawler and Indexer components either alone or in any combination with each other.

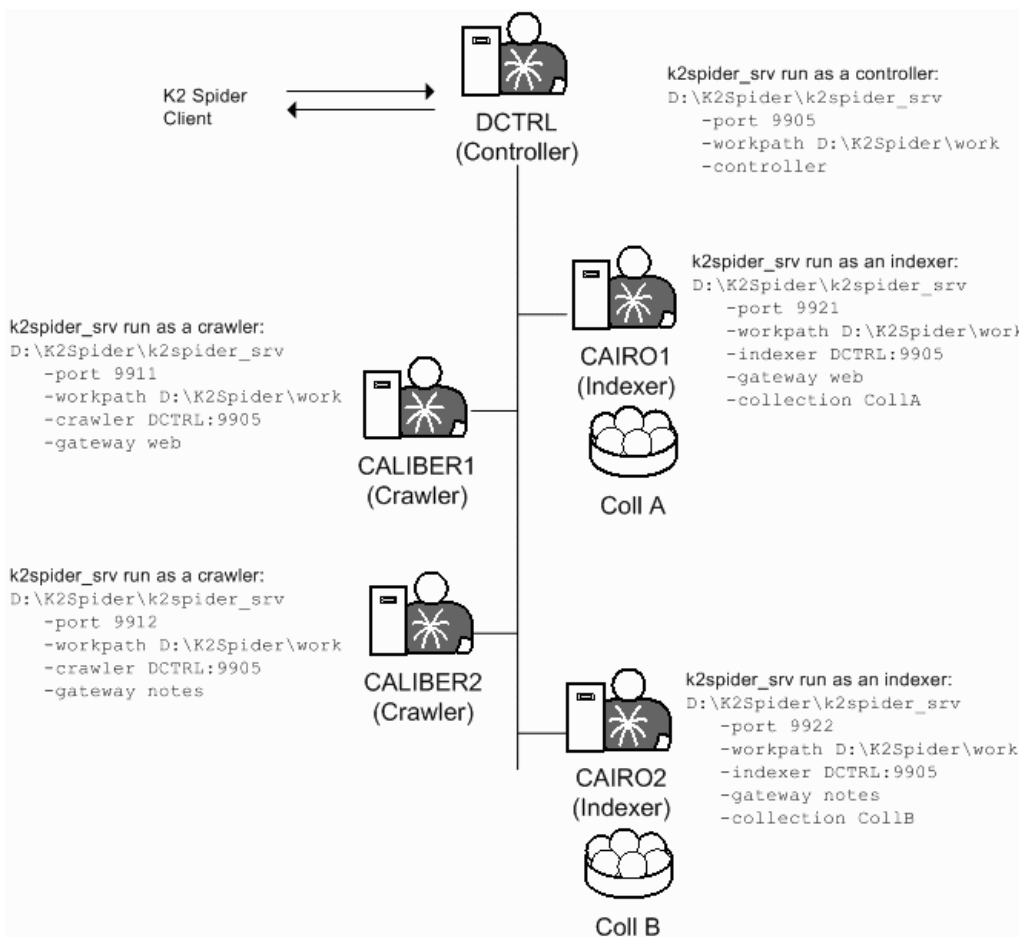
Each Controller can interact with:

- As many as 15 Crawlers
- As many as 15 Indexers
- As many as 10 K2 Spider Clients

K2 Spider System Illustration

The following figure illustrates a distributed K2 Spider system. For more information about other configurations, see [Chapter 6, “Building a K2 Spider System.”](#)

Figure 5-1 A distributed K2 Spider indexing system



Building a K2 Spider System

This chapter describes some of the ways you can build a K2 Spider indexing system.

This chapter includes the following sections:

- [K2 Spider Configurations](#)
- [Binding Crawlers with Indexers](#)
- [Fully Distributing All Components](#)
- [Configuration Considerations](#)

K2 Spider Configurations

K2 Spider indexing system configurations can range from running all of the K2 Spider Server service components in a single process to spreading the workload over multiple machines. The flexibility inherent in the services enables you to distribute the work load across as many machines as necessary to achieve optimum performance. The following are typical configurations:

- [Binding Crawlers with Indexers](#)
- [Fully Distributing All Components](#)
- [Configuration Considerations](#)

Note For information on limitations when combining components, see [“Component Limitations” on page 165](#).

Binding Crawlers with Indexers

For a flexible, distributed system, isolate the Controller to a separate process or server. For fast, efficient indexing, combine Crawlers and Indexers under the same process space.

Binding Tips

There must be a one-to-one relationship between Indexers and collections. For information on limitations when combining components, see [“Component Limitations” on page 165](#).

For document routing, use the `BoundCrwlIndx` router type in the job INI file to take advantage of locating both the Crawler and the Indexer on the same machine. For more information, see [“DocRouter Section” on page 279](#).

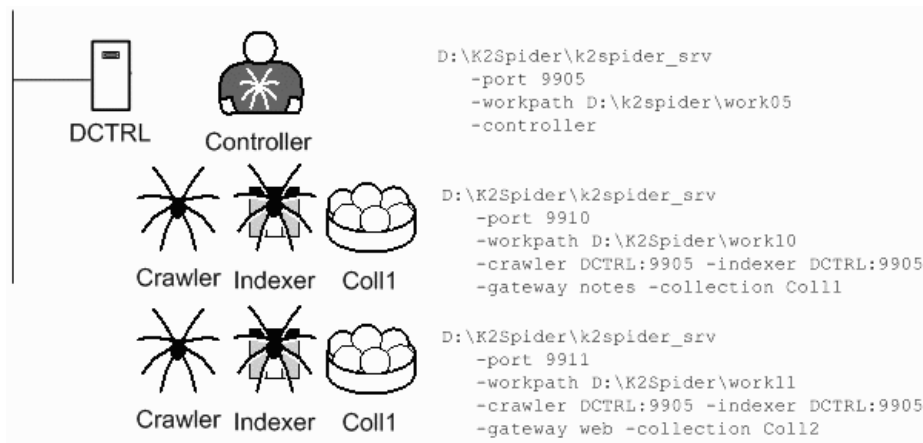
The gateway type of the Crawler and the Indexer must match when indexing Notes, Exchange, ODBC, or Documentum servers. When indexing Web sites or file systems, the gateway type can be `filesys` or `web`.

When the Indexer is run with this gateway type...	The Crawler must be run with this gateway type...
documentum	documentum
exchange	exchange
filesys	web or filesys
notes	notes
odbc	odbc
web	web or filesys

Multiple Processes on a Single Host

To keep your indexing system compact and manageable, while still allowing for flexibility on a single robust host, run multiple instances of the `k2spider_srv` service. Use one `k2spider_srv` instance for the Controller and one additional instance for each pair of Crawlers and Indexers. As the following figure shows, one Crawler/Indexer pair is associated with each collection.

Figure 6-1 Multiple processes on a single host



Matching Gateway Types

The preceding figure shows that by setting the gateway type of the `k2spider_srv` service on port 9910 to `notes`, the `-crawler` and `-indexer` component gateways are also set to `notes`. With the `k2spider_srv` service on port 9911 set to a gateway type of `web`, the `-crawler` and `-indexer` component gateways for that service are also set to `web`.

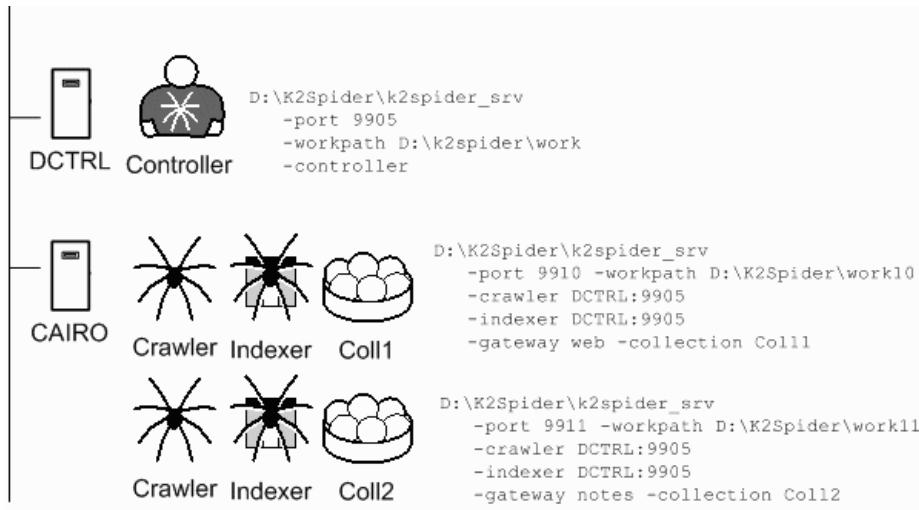
Matching Indexers With Collections

Each Indexer can be matched with only one collection. Specifying the collection can be part of the command when you start the Indexer component. You can also create the collection at a later time by using the K2 Spider Client and the `createcoll` option. In the preceding figure, one `k2spider_srv -indexer` instance is matched with collection `Coll1` and the other `k2spider_srv -indexer` instance is matched with collection `Coll2`.

Separate Controller with One Other Host

If you do not want all of your services on a single host, the configuration shown below places the Controller on its own host with the Crawlers and Indexers on another host. There will be network traffic between the Controller and the other 9host, so make sure you have a high-speed connection that can handle the traffic.

Figure 6-2 Separate Controller with one other host



Matching Gateway Types

The gateway type specified for a collection in the job INI file determines the gateway type for the -crawler and -indexer components for the k2spider_srv service. The preceding figure shows that the gateway type for collection Coll11 is web. This means the gateway type for those -crawler and -indexer components is also web. The gateway type for collection Coll12 is notes, so the gateway type for those -crawler and -indexer components is also notes.

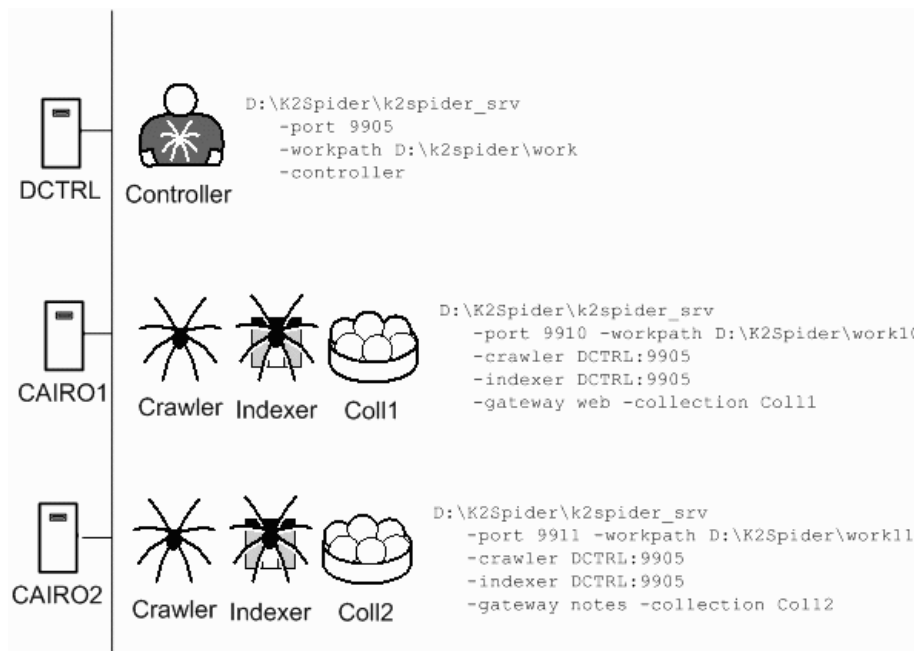
Matching Indexers With Collections

Each Indexer can be paired with only one collection. Specifying the collection can be part of the command when you start the Indexer component. You can also create the collection at a later time by using the K2 Spider Client and the `createcoll` option. In the preceding figure, one `k2spider_srv -indexer` instance is paired with collection `Coll1` and the other `k2spider_srv -indexer` instance is paired with collection `Coll2`. For information on limitations when combining components, see [“Component Limitations” on page 165](#).

Extending the System

If you have frequent updates and refreshes that increase disk activity, you can isolate the workloads further. As the following figure shows, you can distribute each Crawler and Indexer pair to its own server.

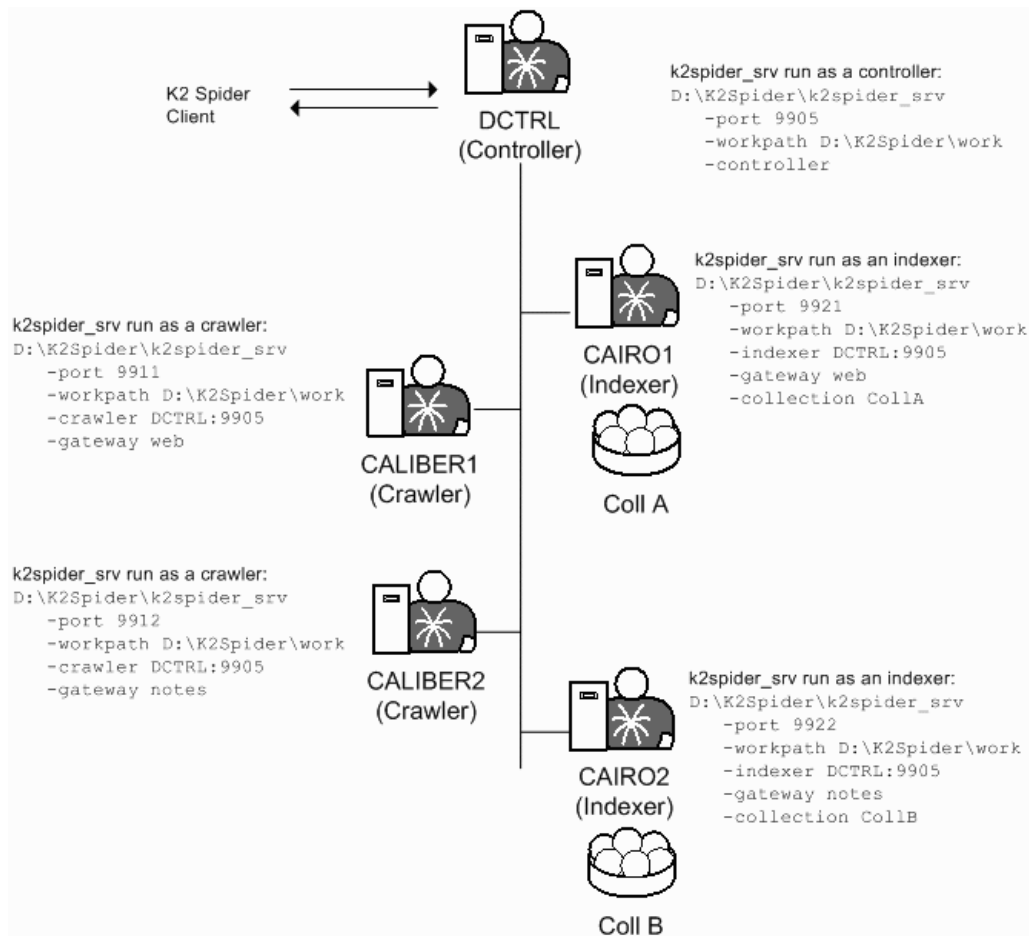
Figure 6-3 Extending a K2 Spider indexing system



Fully Distributing All Components

For the most flexible, distributed system possible, dedicate a host to each instance of the `k2spider_srv` service. This configuration relies on a very fast network connection between the hosts that comprise the entire K2 Spider system. The following figure illustrates a system with fully distributed components.

Figure 6-4 Fully distributing all components



Matching Gateway Types

As the previous figure shows, CALIBER1 and CAIRO1 are paired with a gateway type of `web`, and CALIBER2 and CAIRO2 are paired with a gateway type of `notes`. When you create and run jobs in this configuration, make sure the gateway types match for all of the appropriate items.

Note In the job INI file, you must specify values for the `[Crawlers]` and `[Indexers]` sections, and the components you specify must be of the relevant gateway type.

Web Sites Indexing Jobs

A job indexing Web sites **must** have the following job INI entries which match on the gateway type of `web`.

```
[Job]
gatewaytype = web

[Crawlers]
crawlerspec = CALIBER1:9911

[Indexers]
indexerspec = CAIRO1:9921
```

Notes Server Indexing Jobs

A job indexing Notes Servers must have the following job INI entries which match on the gateway type of `notes`.

```
[Job]
gatewaytype = notes

[Crawlers]
crawlerspec = CALIBER2:9912

[Indexers]
indexerspec = CAIRO2:9922
```

Configuration Considerations

When considering how to distribute the `k2spider_srv` service components and how many to use, consider the following subjects.

Network Connection Considerations

- Network connection speed to data repository servers

The speed of the connection to your data repositories determines how many Crawlers and Indexers you can efficiently use. For example, with slower networks, Indexers may idle while waiting for Crawlers to retrieve documents. For Web site indexing, the Crawler retrieves documents to the local disk cache.

- Network connection speed between all servers

As much as the speed to your data repositories can reduce performance, the speed between your Crawlers or Indexers and the Controller can also hinder the speed and efficiency of your indexing system.

For example, with a very fast network, you can afford to distribute the K2 Spider services because communication between them is fast. On a slow network, however, you may need to consolidate K2 Spider services to reduce network traffic.

Server Resource Considerations

- CPUs, memory and disk speed

On servers with large amounts of memory and multiple CPUs, you can more easily combine Crawlers and Indexers in the same process space to take advantage of the fast communication. For information on limitations when combining components, see [“Component Limitations” on page 165](#).

On servers with multiple fast disks, especially if they also have large amounts of memory and multiple CPUs, you can more easily run multiple Indexers to allow for multiple collections.

Data Repository Considerations

- Web servers

When indexing Web sites, pair Crawlers with Indexers on the same server to reduce network traffic between those components.

- Notes, Exchange, ODBC, and Documentum servers

When indexing Notes, Exchange, ODBC, and Documentum servers, use only one Crawler for each data repository server.

- File systems

When you index unified file systems that are distributed across remote servers and you use duplicate detection (`detectdupfile = true`), run a Crawler and an Indexer on each remote server being indexed.

Take into account that on Windows, hidden and system files are indexed. If you do not want such files included in an indexing job, use the appropriate criteria to exclude them.

Running K2 Spider Server

This chapter describes how to use the components (Controller, Crawler, Indexer) of the K2 Spider Server service, `k2spider_srv`.

Note You cannot use different versions of K2 Spider Server (`k2spider_srv`) and K2 Spider Client (`k2spider_cli`) with each other. K2 Spider Server and K2 Spider Client must be the same version.

This chapter includes the following sections:

- [Running a Controller](#)
- [Running a Crawler](#)
- [Running an Indexer](#)
- [k2spider_srv Reference](#)

Running a Controller

The Controller is the heart of a K2 Spider system. K2 Spider Client requests, including indexing jobs and their accompanying initialization files, are passed to the Controller which then decides how to manage them.

Using k2spider_srv as a Controller

To run the k2spider_srv service as a Controller, use the following command:

```
k2spider_srv -controller [options]
```

Note When you do not specify a port number with the -port option, k2spider_srv defaults to port 8080.

Options for Using k2spider_srv as a Controller

The supported options for the k2spider_srv service when it is run as a Controller are described below. The **Type** indicator is used to specify unique usage requirements. For example, the entry may indicate that the option is required or that the option is used only with a particular gateway type.

-cmdfile

Description Specifies that k2spider_srv reads command line syntax from a file in addition to the options passed from the command line.

Syntax -cmdfile *path_and_filename*

Details This option can also be used by a Crawler and an Indexer. This option includes the path name to the file containing the command line options. The -cmdfile option circumvents command line length limits.

The syntax for the text file that holds the commands is:

option option_parameters

For better readability, put each option and any parameters on a single line. k2spider_srv properly parses the lines.

-controller

Description Specifies that the `k2spider_srv` runs as a Controller on the specified port.

-help

Description Displays the `k2spider_srv` options.

Note Running `k2spider_srv` without any options displays the list of options.

This option can also be used by a Crawler and an Indexer.

-ip

Description Specifies an IP address to which to bind this instance of `k2spider_srv`.

Syntax `-ip nnn.nnn.nnn.nnn`

Details This option can also be used by a Crawler and an Indexer. Use this option when you have multiple network interface cards or specific DNS requirements. When you start an instance of `k2spider_srv` with this option, you must specify the corresponding IP address and not the host name when connecting to this `k2spider_srv` with `k2spider_cli`.

-license

Description Loads the license by specifying the path to the K2 Spider license file.

Syntax `-license path`

Default The license resides within:

`installDir/k2/common/runtime.lic`

where `installDir` is the directory in which you installed K2 Spider.

WARNING! The `runtime.lic` file must be in a `/common` directory regardless of where it resides. For example, if you wanted to store the file in `/user/apps/licenses`, then you would need the following: `/user/apps/licenses/common/runtime.lic`.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (/). Examples in Windows notation uses backward slashes (\).

Note This option can also be used by a Crawler and an Indexer.

-loglevel

Description Specifies the level of logging.

Syntax `-loglevel skip|summary|verbose | debug | trace`

Details Select only one value from `skip`, `summary`, `verbose`, `debug`, or `trace`. This option specifies the maximum level of logging available to indexing jobs. Messages written to the log file are equal to or lower in verbosity than the level specified with `loglevel` in a job INI file. For example, if you specify `-loglevel summary`, then `verbose`, `debug` and `trace` level messages are not included.

Note K2 Spider does not allow its log files to grow beyond 500 MB. When a log file reaches 500 MB, it is backed up as `FileName.old`. When the new file grows to 500 MB, it is renamed to `FileName.old`, overwriting the previous `FileName.old`. Thus an administrator can see only the most recent 500MB - 1 GB of log data.

This option can also be used by a Crawler and an Indexer.

Default By default, only process related error messages are logged.

See Also `loglevel` in [Chapter 9, "Job Initialization File Reference."](#)

-msgdb

Description Specifies the path to the `k2spider.msg` message database file.

Syntax `-msgdb path`

Default The `k2spider.msg` message database is read from:

`installDir/k2/common/k2spider.msg`

where `installDir` is the directory in which you installed K2 Spider.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (/). Examples in Windows notation uses backward slashes (\).

Note This option can also be used by a Crawler and an Indexer.

-output

Description Specifies the path and file name for output from stdout and stderr.

Syntax `-output path_and_filename`

Details When running K2 Spider from Java code, it may sometimes hang. This is caused by issues with handling stdout and stderr. Use the `-output` option to force K2 Spider to redirect all stdout and stderr messages to the specified file.

Note This option can also be used by a Crawler and an Indexer.

-port

Description Specifies the port number for the `k2spider_srv` service running as a Controller.

Syntax `-port port_num`

Default When you do not specify a port number with the `-port` option, `k2spider_srv` defaults to port 8080.

Note This option can also be used by a Crawler and an Indexer.

-recover

Description Specifies that the Controller should run any jobs which were left unfinished by any interruption.

Details By default, when you bring the Controller back up after an interruption, unfinished jobs are not resumed automatically.

If you do not use the `-recover` option, you must manually re-start unfinished jobs using the K2 Spider Client `start` option. For more information, see `start` in [Chapter 8, “Using K2 Spider Client.”](#)

Note When you specify the `-recover` option, you must also specify the original work path with the `-workpath` option.

-workpath

Description Specifies a working directory where the Controller may store databases, configuration and temporary files.

Syntax `-workpath path`

Details By default, the *path* is the current working directory from which `k2spider_srv` was run. The directories and their contents are:

- *path* — The *path* directory contains the `status.log` file. Examples of messages written to `status.log` are: running out of disk space; keys not being sent to a Crawler or Indexer; a Crawler or Indexer failing to send feedback to the Controller. Messages in `status.log` are not tied to any particular job.
- *ini* — The *path\ini* sub-directory contains the `job.ini` files which were created on the Controller. The `k2ssrv.cfg` configuration file contains information used by the Controller. As you use the K2 Spider Client to create each job, a copy of that `job.ini` file is placed into this directory. Do not edit these files manually.
- *temp* — The *path\temp* sub-directory contains temporary files during indexing jobs and is always cleared once an indexing job is completed. You can also specify the *temp* directory by using the `-temp` option.

For the Controller component, the *temp* directory exists, but does not contain anything.

- *jobname* — The *path\jobname* sub-directory contains job-related files and databases as well as the following sub-directories. The value of *jobname* corresponds to the indexing job name.
 - The *path\jobname\vsdb* sub-directory contains the K2 Spider database files for internal use by the Controller.
 - The *path\jobname\log* sub-directory contains all of the indexing job log files:
 - `skipkeys.log` — Created per job, only when log level is `skip` or higher and any keys are actually skipped. This file is only created for a Controller.
 - `badkeys.log` — Created per job, only when bad keys are actually encountered. This file is only created for a Controller.
 - `jobstatus.log` — Created per `k2spider_srv` process, per job, only when log level for the job is set in the job INI file to `summary`, `verbose`, `debug` or `trace`. This log file is not created by default, as the default log level is `skip`.

`joberror.log` — Created per `k2spider_srv` process, per job. If there are errors during processing on the Controller, the messages appear here. If there are errors during processing on a Crawler or Indexer, the messages appear in the `joberror.log` file in the work path for that `k2spider_srv` process running as a Crawler or Indexer.

For information on indexing job logging, see `loglevel` in [Chapter 9, “Job Initialization File Reference.”](#)

These directories are created for you beneath the last directory specified in *path*.

Default The *path* is the working directory from which `k2spider_srv` was run.

Note This option can also be used by a Crawler and an Indexer.

Warning! Whenever you start a Controller, make sure the work path, as specified by `-workpath`, is clean. If there are any configuration files in the path, the Controller will read them and use the Crawlers and Indexers specified there, rather than those specified in any `job.ini` files. Also ensure that no two `k2spider_srv` processes share the same workpath. Each `k2spider_srv` process must have a unique value for `-workpath`.

Running a Crawler

The Crawler is responsible for gathering the documents to be indexed.

Using k2spider_srv as a Crawler

To run the `k2spider_srv` service as a Crawler, use the following command:

```
k2spider_srv -crawler controller-spec -port portnum  
            -workpath path [options]
```

Note If you have more than one Crawler with a gateway type of Notes, Exchange, ODBC, or Documentum, each Crawler instance must have a unique value for the `-workpath` option.

When you do not specify a port number with the `-port` option, `k2spider_srv` defaults to port 8080.

Sharing the Process Space

You can share the process space used by the `-crawler` option with the `-indexer` option. Note that the `status.log`, `jobstatus.log` and `joberror.log` files are written per `k2spider_srv` process, not per component (Controller, Crawler, Indexer).

Gathering Relevance Information

When serving an HTTP gateway-based collection, where a Web server is indexed, the Crawler must first traverse the entire Web site before indexing can begin by the Indexer. How much of the site is traversed is dependent on the indexing job's criteria, as defined in the job INI file.

All of the links that will be available to the Indexer must be found in order to calculate relevancy information.

Options for Using k2spider_srv as a Crawler

The supported options for the k2spider_srv service when it is run as a Crawler are described below. The **Type** indicator is used to specify unique usage requirements.

-anonymous

Description Specifies that the Crawler component is not to prompt for credentials for supplemental filtering when crawling.

Type Notes only (`-gateway` notes)

Note This option can also be used by an Indexer.

See Also *Verity K2 Lotus Notes Gateway Guide*

-certfile

Description Specifies the path to the SSL client certificate to be used for indexing SSL-secured web sites which require client certificates.

Type This option is required with `-keyfile`.

Syntax `-certfile path_and_filename`

Details The format of the certificate must be in the form of an X.509 certificate.

-cmdfile

Description Specifies that k2spider_srv reads command-line syntax from a file in addition to the options passed from the command-line.

Syntax `-cmdfile path_and_filename`

Details This option can also be used by a Controller and an Indexer. This option includes the path name to the file containing the command-line syntax. The `-cmdfile` option circumvents command-line length limits.

The syntax for the text file which holds the commands is:

option option_parameters

For better readability, put each option and any parameters on a single line. k2spider_srv properly parses the lines.

-connections

Description Specifies the maximum number of simultaneous socket connections to make to data repositories such as web sites for crawling.

Syntax `-connections num_connections`

Details Each connection implies a separate thread.

Default 6 connections.

Note The dynamic flow control of the `k2spider_srv` service makes the most use of all available connections when indexing web sites. If you are indexing multiple sites, you may want to increase this number. Increasing the number of connections may not always help because of such dependencies as your CPU, memory and network connection speed and the capabilities of the sites targeted for indexing.

-crawler

Description Specifies that the `k2spider_srv` service will run as a Crawler governed by the specified Controller.

Syntax `-crawler controller_host:port`

-gateway

Description Specifies the appropriate Verity gateway to be used to index the repository.

Syntax `-gateway gw_type`

Details Valid values are `notes`, `odbc`, `documentum`, `exchange`, `filesys`, `web`, `gateway`. The defaults are `filesys` and `web`.

The gateway value for `gw_type` is reserved for OEM customers using the Gateway Developers Kit (GDK). For more information, see the *Verity OEM Deployment Guide*.

Note The value you specify for `-gateway` must match the value for `-gateway` for any Indexers used by this Crawler, and it must match the value for `gatewaytype` in all jobs processed by this Crawler. For more information, see `gatewaytype` in [Chapter 9, "Job Initialization File Reference."](#)

This option can also be used by an Indexer.

-gwauth

Description Specifies a file to support authentication for secure repositories.

Type For Notes, ODBC, Documentum and Exchange only (`-gateway notes`, `-gateway odbc`, `-gateway documentum` or `-gateway exchange`).

Syntax `-gwauth path_and_filename`

Note This option can also be used by an Indexer.

See Also `auth` entry in [Chapter 9, “Job Initialization File Reference.”](#)

-help

Description Displays the `k2spider_srv` options. This option can also be used by a Controller and an Indexer.

Note Running `k2spider_srv` without any options displays the list of options.

-https

Description Enables the indexing of SSL-enabled web sites when you have the Verity SSL Option installed.

Type Web sites only (`-gateway web`)

-ip

Description Specifies an IP address to which to bind this instance of `k2spider_srv`.

Syntax `-ip nnn.nnn.nnn.nnn`

Details This option can also be used by a Controller and an Indexer. Use this option when you have multiple network interface cards or specific DNS requirements. When you start an instance of `k2spider_srv` with this option, you must specify the corresponding IP address and not the host name when connecting to this `k2spider_srv` with `k2spider_cli`.

-keyfile

Description Specifies the path to the private key file to be used with `-certfile`.

Syntax `-keyfile path_and_filename`

-license

Description Loads the license by specifying the path to the K2 Spider license file.

Syntax `-license path`

Default The license resides within `installDir/k2/common/runtime.lic`, where `installDir` is the directory in which you installed K2 Spider.

WARNING! The `runtime.lic` file must be in a `/common` directory regardless of where it resides. For example, if you wanted to store the file in `/user/apps/licenses`, then you would need the following: `/user/apps/licenses/common/runtime.lic`.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (`/`). Examples in Windows notation uses backward slashes (`\`).

Note This option can also be used by a Controller and an Indexer.

-loglevel

Description Specifies the level of logging.

Syntax `-loglevel skip|summary|verbose | debug | trace`

Details Select only one value from `skip`, `summary`, `verbose`, `debug`, or `trace`. This option specifies the maximum level of logging available to indexing jobs. Messages written to the log file are equal to or lower in verbosity than the level specified with `loglevel` in a job INI file. For example, if you specify `-loglevel summary`, then `verbose`, `debug` and `trace` level messages are not included.

Note K2 Spider does not allow its log files to grow beyond 500 MB. When a log file reaches 500 MB, it is backed up as `FileName.old`. When the new file grows to 500 MB, it is renamed to `FileName.old`, overwriting the previous `FileName.old`. Thus an administrator can see only the most recent 500MB - 1 GB of log data.

This option can also be used by a Controller and an Indexer.

Default By default, only process related error messages are logged.

See Also `loglevel` in [Chapter 9, "Job Initialization File Reference."](#)

-msgdb

Description Specifies the path to the `k2spider.msg` message database file.

Syntax `-msgdb path`

Default The `k2spider.msg` message database is read from:

`installDir/k2/common/k2spider.msg`

where `installDir` is the directory in which you installed K2 Spider.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (/). Examples in Windows notation uses backward slashes (\).

Note This option can also be used by a Controller and an Indexer.

-output

Description Specifies the path and file name for output from stdout and stderr.

Syntax `-output path_and_filename`

Details When running K2 Spider from Java code, it may sometimes hang. This is caused by issues with handling stdout and stderr. Use the `-output` option to force K2 Spider to redirect all stdout and stderr messages to the specified file.

Note This option can also be used by a Controller and an Indexer.

-port

Description Specifies the port number for the `k2spider_srv` service running as a Crawler.

Syntax `-port port_num`

Default When you do not specify a port number with the `-port` option, `k2spider_srv` defaults to port 8080.

When you combine a Crawler with an Indexer, only one port number is required:

```
k2spider_srv -indexer -crawler -port 9191
```

When you combine a Crawler and an Indexer, the respective specifications in a job INI file would look like this:

```
[Crawlers]
crawlerspec = cairo:9191
[Indexers]
indexerspec = cairo:9191
```

Note This option can also be used by a Controller and an Indexer.

-style

Description Specifies the path to the style files to use when creating a new collection.

Type For all gateways

Syntax `-style path`

Details When specifying `-style`, you must use the style files appropriate for each indexing gateway.

Style files created with the K2 Dashboard are located in:

`dataDir/stylesets/stylesetName`

where *dataDir* is the data directory.

For information on the K2 Dashboard and creating style sets, see the *Verity K2 Dashboard Administrator Guide* and the relevant gateway guide if necessary.

You can create style sets for the Lotus Notes Gateway, the Documentum Gateway, and the Exchange Gateway using the StyleSet Editor in standalone mode. See the relevant gateway guide for more information on where the style sets can be created.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (/). Examples in Windows notation uses backward slashes (\).

-temp

Description Specifies the directory for temporary files (disk cache).

Syntax `-temp path`

Default The `temp` directory is contained within the `workpath` directory structure as defined by `-workpath`.

Note Make sure the location you specify contains enough disk space to handle the documents which are downloaded and held before indexing. The documents are deleted from the `temp` directory after they are indexed.

This option can also be used by an Indexer.

-workpath

Description Specifies a working directory where the Crawler may store databases, configuration and temporary files.

Type This option is required for a Crawler.

Syntax `-workpath path`

Details The directories and their contents are:

- *path* — The *path* directory contains the `status.log` file. Examples of messages written to `status.log` are: running out of disk space; keys not being sent to a Crawler or Indexer; a Crawler or Indexer failing to send feedback to the Controller. Messages in `status.log` are not tied to any particular job.
- *ini* — The *path*\ini subdirectory is empty when it is first created. As you use the K2 Spider Client to create each job, a copy of that `job.ini` file is placed into this directory. Do not edit these files manually.
- *temp* — The *path*\temp sub-directory contains temporary files during indexing jobs and is always cleared once an indexing job is completed. You can also specify the temp directory by using the `-temp` option.
- *jobname* — The *path*\jobname sub-directory contains job-related files and databases as well as the following sub-directories. The value of *jobname* corresponds to the indexing job name.
 - The *path*\jobname\log sub-directory contains all of the indexing job log files:
 - `jobstatus.log` — Created per `k2spider_srv` process, per job, only when log level for the job is set in the job INI file to summary, verbose, debug or trace. This log file is not created by default, as the default log level is skip.
 - `joberror.log` — Created per `k2spider_srv` process, per job. If there are errors during processing on a Crawler, the messages appear here. If there are errors during processing on the Controller, the messages appear in the `joberror.log` file in the work path for that `k2spider_srv` process running as a Controller.
 - For information on indexing job logging, see `loglevel` in [Chapter 9, “Job Initialization File Reference.”](#)

These directories are created for you beneath the last directory specified in *path*.

Default The *path* is the working directory from which `k2spider_srv` was run.

Note If you have more than one Crawler with a gateway type of Notes, Exchange, ODBC, or Documentum, each Crawler instance must have a unique `-workpath` value.

Also ensure that no two `k2spider_srv` processes share the same workpath. Each `k2spider_srv` process must have a unique `-workpath` value.

This option can also be used by a Controller and an Indexer.

Running an Indexer

The Indexer is responsible for indexing documents into a collection.

Using k2spider_srv as an Indexer

To run the k2spider_srv service as an Indexer, use the following command:

```
k2spider_srv -indexer controller-spec -port portnum  
            -workpath path [options]
```

Note When you do not specify a port number with the `-port` option, k2spider_srv defaults to port 8080.

Sharing the Process Space

You can share the process space used by the `-indexer` option with the `-crawler` option. Note that the `status.log`, `jobstatus.log` and `joberror.log` files are written per k2spider_srv process, not per component (Controller, Crawler, Indexer).

Options for Using k2spider_srv as an Indexer

The supported options for the k2spider_srv service when it is run as an Indexer are described below. The **Type** indicator is used to specify unique usage requirements.

-anonymous

Description Specifies that the k2spider_srv -indexer service is not to prompt for credentials for supplemental filtering when indexing.

Type Notes only (`-gateway` notes)

Note This option can also be used by a Crawler.

See Also *Verity K2 Lotus Notes Gateway Guide*

-charmap

Description Specifies the character map to use depending on the value for `-locale`.

Syntax `-charmap charset_name`

Details Use the character set specified in `charset_name` to display the contents of the collection. `charset_name` must be the name of one of the supported character sets for the collection's locale.

This option is not required if you want to display the collection data using its locale's internal character set.

Appendix A of the *Verity Locale Configuration Guide* lists the supported character sets for each Verity locale and indicates which one is the internal character set.

See Also For information on supported locales and their use, see the *Verity Locale Configuration Guide*.

-cmdfile

Description Specifies that `k2spider_srv` reads command-line options from a file in addition to using the options passed from the command-line.

Syntax `-cmdfile path_and_filename`

Details This option can also be used by a Controller and a Crawler. This option includes the path name to the file containing the command-line syntax. The `-cmdfile` option circumvents command-line length limits.

The syntax for the text file which holds the commands is:

option option_parameters

For better readability, put each option and any parameters on a single line. `k2spider_srv` properly parses the lines.

-collection

Description Specifies the full path to the collection you want to create or update.

Type This option is required.

Syntax `-collection path_and_filename`

Details If the collection does not exist, it will be created using the path you specify.

-common

Description Specifies the path to the Verity home directory, *installDir/k2/common*, where *installDir* is the directory in which you installed K2 Spider.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward slashes (/). Examples in Windows notation uses backward slashes (\).

Syntax `-common path_to_common_dir`

Note This option is typically not needed, as long as the PATH environment variable is set correctly during installation.

-datefmt

Description Specifies the Verity import date format to use. Valid values are MDY, DMY, YMD, USA and EUR.

Syntax `-datefmt format`

Default MDY

-gateway

Description Specifies the Verity gateway to be used to index the appropriate repository type.

Syntax `-gateway gw_type`

Details Valid values are *notes*, *odbc*, *documentum*, *exchange*, *filesystem*, *web*, *gateway*. The defaults are *filesystem* and *web*.

The gateway value for *gw_type* is reserved for OEM customers using the Gateway Developers Kit (GDK). For more information, see the *Verity OEM Deployment Guide*.

Note The value you specify for `-gateway` must match the value for `-gateway` for any Crawlers used by this Indexer, and it must match the value for *gatewaytype* in all jobs processed by this Indexer.

This option can also be used by a Crawler.

See Also *gatewaytype* in [Chapter 9, "Job Initialization File Reference."](#)

-gwauth

Description Specifies an authorization file to support authentication for secure repositories.

Type For Notes, Exchange, Documentum, and ODBC only (`-gateway notes`, `-gateway exchange`, `-gateway documentum`, or `-gateway odbc`)

Syntax `-gwauth path_and_filename`

Note This option can also be used by a Crawler.

See Also `auth` entry in [Chapter 9, “Job Initialization File Reference.”](#)

-help

Description Displays the `k2spider_srv` options.

Note Running `k2spider_srv` without any options displays the list of options.

This option can also be used by a Controller and a Crawler.

-indexer

Description Specifies the `k2spider_srv` service runs as a Indexer governed by the specified Controller.

Syntax `-indexer controller_host:port`

-ip

Description Specifies an IP address to which to bind this instance of `k2spider_srv`.

Syntax `-ip nnn.nnn.nnn.nnn`

Details This option can also be used by a Controller and a Crawler. Use this option when you have multiple network interface cards or specific DNS requirements. When you start an instance of `k2spider_srv` with this option, you must specify the corresponding IP address and not the host name when connecting to this `k2spider_srv` with `k2spider_cli`.

-indxsessions

Description Specifies the maximum number of indexing threads to run on a collection.

Syntax `-indxsessions num_sessions`

Default 2 indexing threads. Increasing the value of *num_sessions* requires additional CPU and memory resources.

See Also `-maxindmem`

Note The Indexer component (`k2spider_srv -indexer`) is hard-coded to use a single indexing thread when indexing PDFs.

-license

Description Loads the license by specifying the path to the K2 Spider license file.

Syntax `-license path`

Default The license resides within *installDir*/k2/common/runtime.lic, where *installDir* is the directory in which you installed K2 Spider.

WARNING! The `runtime.lic` file must be in a `/common` directory regardless of where it resides. For example, if you wanted to store the file in `/user/apps/licenses`, then you would need the following: `/user/apps/licenses/common/runtime.lic`.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward (/) slashes. Examples in Windows notation uses backward (\) slashes.

Note This option can also be used by a Controller and a Crawler.

-locale

Description Specifies the Verity locale to use for indexing.

Syntax `-locale locale_name`

Details The name of the locale in which the collection you are accessing was created. `locale_name` must be the name of a locale for which you are licensed, and must be one of the Verity locales listed in Appendix A of the *Verity Locale Configuration Guide*.

This option is not required if the collection or index uses the default session locale (usually `uni`).

Default The default value is installation-dependent.

See Also For more information on supported locales and their use, see the *Verity Locale Configuration Guide*.

-loglevel

Description Specifies the level of logging.

Syntax `-loglevel skip|summary|verbose | debug | trace`

Details Select only one value from `skip`, `summary`, `verbose`, `debug`, or `trace`. This option specifies the maximum level of logging available to indexing jobs. Messages written to the log file are equal to or lower in verbosity than the level specified with `loglevel` in a job INI file. For example, if you specify `-loglevel summary`, then `verbose`, `debug` and `trace` level messages are not included.

Note K2 Spider does not allow its log files to grow beyond 500 MB. When a log file reaches 500 MB, it is backed up as `FileName.old`. When the new file grows to 500 MB, it is renamed to `FileName.old`, overwriting the previous `FileName.old`. Thus an administrator can see only the most recent 500MB - 1 GB of log data.

This option can also be used by a Controller and a Crawler.

Default By default, only process related error messages are logged.

See Also `loglevel` in [Chapter 9, “Job Initialization File Reference.”](#)

-maxindfilehandles

Description When running K2 Spider on UNIX, specifies the maximum number of file handles of memory, in kilobytes, to be used by each indexing thread.

Syntax `-maxindfilehandles n`

Details The number of file handles per VDK session started by an Indexer.

Default The default value is $(ulimit -256) / (\text{number of indexer sessions})$.

Note Use this option with the help of Verity Technical Support.

-maxindmem

Description Specifies the maximum amount of memory, in kilobytes, to be used by each indexing thread.

Syntax `-maxindmem kilobytes`

Details The number of threads is specified with `-indxsessions`.

Default The amount of memory assigned for each indexing thread is determined by the following:

The amount of free system memory is divided by the number of indexing threads, with a minimum of 8 MB per thread.

-msgdb

Description Specifies the path to the `k2spider.msg` message database file.

Syntax `-msgdb path`

Default The `k2spider.msg` message database is read from:

`installDir/k2/common/k2spider.msg`

where *installDir* is the directory in which you installed K2 Spider.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward (/) slashes. Examples in Windows notation uses backward (\) slashes.

Note This option can also be used by a Controller and a Crawler.

-noindex

Description Specifies that the Indexer component is to gather document locations which it stores in a bulk insert file (BIF), and then submit the BIF to the collection for indexing the next time an indexing service is run for the collection.

Details Keep in mind that the work, in the form of the BIF, is still submitted to the collection for eventual indexing. However, the work is just not done in the same process space and time as the Indexer component used with `-noindex`. If you want to gather document locations into a BIF without submitting it to the collection at all, use the `nosubmit` option in the job's INI file. Using `nosubmit` will allow you to decide exactly when to submit a follow-up indexing service to work on the BIF.

The `-noindex` option is typically used in conjunction with a separate indexing service, such as `mkvdk`. The BIF will be processed by the next indexing service run for the collection, whether it is the Indexer component or `mkvdk`.

Do not try to start both `k2spider_srv` and another indexing service at the same time. You must allow `k2spider_srv` enough time to generate enough work for the secondary indexing service. If you are using `mkvdk`, you can run it in persistent mode to ensure it will act upon work generated by `k2spider_srv`.

Warning! When you execute an indexing job for a collection and you use `-noindex`, the persistent store for the collection is not updated. Furthermore, the Indexer component will be in no-index mode for the entire time it is running. You should not use any of the K2 Spider Client `submit item` options while a Indexer component is running in no-index mode.

See Also `nosubmit` in [Chapter 9, "Job Initialization File Reference."](#)

-nooptimize

Description Prevents the `k2spider_srv` service from optimizing the collection, thus reducing processing overhead during the indexing job.

Details You should use this option sparingly, as it leaves the collection in less-than-optimal condition. Furthermore, the Indexer component will be in `-nooptimize` mode for the entire time it is running. Some examples of when you might want to use this option are:

- You want to manually perform custom optimization of the collection, using `mkvdk`. By default, Indexer optimization mimics the `mkvdk` actions of `maxmerge`, `maxclean` and `vdbopt`. For more information on `mkvdk`, see the *Verity Collection Reference*.
- You are running multiple, sequential indexing jobs against a collection, and want to wait until they are all finished to optimize.

Generally, do not leave a collection unoptimized, as search times can slow significantly. In brief, optimizing a collection means creating a small number of large partitions, which can greatly reduce search times.

-numpages

Description Specifies the VDK session disk cache size for each VDK session created by this Indexer. The number of sessions can be set by using the `-indxsessions` option.

Syntax `-numpages num`

Details The VDK session disk cache size.

Note Use this option with the help of Verity Technical Support.

See Also `-indxsessions`

-output

Description Specifies the path and file name for output from stdout and stderr.

Syntax `-output path_and_filename`

Note This option can also be used by a Controller and a Crawler.

-port

Description Specifies the port number for the `k2spider_srv` service running as a Indexer.

Syntax `-port port_num`

Default When you do not specify a port number with the `-port` option, `k2spider_srv` defaults to port 8080.

When you combine an Indexer with a Crawler, you only need to specify one port number. For example:

```
k2spider_srv -crawler -indexer -port 9191
```

The above example command is not a complete command. You may require more options for a complete command.

When you combine an Indexer and a Crawler, the respective specifications in a job INI file should look like:

```
[Crawlers]
crawlerspec = cairo:9191

[Indexers]
indexerspec = cairo:9191
```

Note This option can also be used by a Controller and a Crawler.

-style

Description Specifies the path to the style files to use when creating a new collection.

Type For all gateways

Syntax `-style path`

Details When specifying `-style`, you must use the style files appropriate for each indexing gateway. Style files created with the K2 Dashboard are located in:

`dataDir/stylesets/stylesetname`

where `dataDir` is the data directory.

For information on the K2 Dashboard and creating style sets, see the *Verity K2 Dashboard Administrator Guide* and the relevant gateway guide if necessary.

You can create style sets for the Lotus Notes Gateway, the Documentum Gateway, and the Exchange Gateway using the StyleSet Editor in standalone mode. See the relevant gateway guide for more information on where the style sets can be created.

Because K2 Spider runs on both UNIX and Windows platforms, the examples may use either notation. The example above is in UNIX notation and uses forward (/) slashes. Examples in Windows notation uses backward (\) slashes.

Note You can safely omit `-style` when resubmitting an indexing job for the same collection, as the style files will become part of the collection.

-submitsize

Description Specifies the number of documents concurrently submitted for indexing.

Syntax `-submitsize num_documents`

Details The upper limit is 64,000.

Although larger values may mean more efficient processing by the Indexer, smaller values allow more parallelism on multi-CPU systems. Furthermore, in the event of a halt during indexing, a smaller value means fewer documents will be lost.

If a halt occurs during indexing, the chunk of documents specified by `-submitsize` is lost because there is no transactional rollback for indexing and the documents are no longer in the queue for indexing. Remember that when you rerun the indexing job, `k2spider_srv` can only continue with URLs and queued documents.

Default 1024 documents.

-temp

Description Specifies the directory for temporary files (disk cache).

Syntax `-temp path`

Default The temp directory is contained within the workpath directory structure as defined by `-workpath`.

Note Make sure the location you specify contains enough disk space to handle the documents which are downloaded and held before indexing. The documents are deleted from the temp directory after they are indexed.

This option can also be used by a Crawler.

See Also `-workpath`

-vdkclean

Description Specifies the threshold number of documents during indexing at which the VDK session is cleaned up.

Syntax `-vdkclean N`

Default 10,000 documents.

Note Use this option with the help of Verity Technical Support.

-workpath

Description Specifies a working directory where the Indexer may store databases, configuration and temporary files.

Type This option is required for an Indexer.

Syntax `-workpath path`

Details The directories and their contents are:

- *path* — The *path* directory contains the `status.log` file. Examples of messages written to `status.log` are: running out of disk space; keys not being sent to a Crawler or Indexer; a Crawler or Indexer failing to send feedback to the Controller. Messages in `status.log` are not tied to any particular job.
- *ini* — The *path\ini* sub-directory is empty.

- **temp** — The *path\temp* sub-directory contains temporary files during indexing jobs and is always cleared once an indexing job is completed. You can also specify the temp directory by using the `-temp` option.
- **submit** — The *path\submit* sub-directory is empty.
- **jobname** — The *path\jobname* sub-directory contains job-related files and databases as well as the following sub-directories. The value of *jobname* corresponds to the indexing job name.
 - The *path\jobname\log* sub-directory contains all of the indexing job log files:
 - jobstatus.log* — Created per *k2spider_srv* process, per job, only when log level for the job is set in the job INI file to *summary*, *verbose*, *debug* or *trace*. This log file is not created by default, as the default log level is *skip*.
 - joberror.log* — Created per *k2spider_srv* process, per job. If there are errors during processing on an Indexer, the messages appear here. If there are errors during processing on the Controller, the messages appear in the *joberror.log* file in the work path for that *k2spider_srv* process running as a Controller.
 - For information on indexing job logging, see *loglevel* in [Chapter 9, “Job Initialization File Reference.”](#)

These directories are created for you beneath the last directory specified in *path*.

Default The *path* is the working directory from which *k2spider_srv* was run.

Note If you have more than one Indexer with a gateway type of Notes, Exchange, ODBC, or Documentum, each Indexer instance must have a unique `-workpath` value.

Also ensure that no two *k2spider_srv* processes share the same workpath. Each *k2spider_srv* process must have a unique value for `-workpath`.

This option can also be used by a Controller and a Crawler.

k2spider_srv Reference

The following table summarizes the output for the `-help` option of `k2spider_srv`. The Components column identifies the components which can use the command.

Usage	Components	Description
<code>-anonymous</code>	CRWL ---- INDX	Specifies the use of the Notes anonymous user for Notes Gateway Indexing.
<code>-gwauth filepath</code>	CRWL ---- INDX	Specifies path to ASCII text file containing repository credentials.
<code>-charmap name</code>	---- ---- INDX	Specifies the collection's character set.
<code>-certfile filepath</code>	CRWL ---- ----	Specifies the path to the SSL Client Certificate (use with <code>-keyfile</code>).
<code>-cmdfile filepath</code>	CRWL CTRL INDX	Specifies the path to the ASCII text file that contains <code>k2spider_srv</code> options.
<code>-collection dirpath</code>	---- ---- INDX	Specifies the collection to create/use.
<code>-common dirpath</code>	---- ---- INDX	Specifies path to the VDK common directory.
<code>-connections num</code>	CRWL ---- ----	Specifies the number of concurrent connections [4].
<code>-controller</code>	---- CTRL ----	Starts a Controller component.
<code>-crawler c_host:port</code>	CRWL ---- ----	Starts a Crawler component and registers it under the specified Controller.
<code>-datefmt fmt</code>	---- ---- INDX	Specifies the collection's date input format.
<code>-gateway gwtype</code>	CRWL ---- INDX	Specifies the Verity gateway type used by a Crawler (<code>-crawler</code>) and/or Indexer (<code>-indexer</code>). Choose from one of the following: notes odbc documentum exchange filesys web gateway For more information about the gateway argument for OEM customers using the Gateway Developers Kit (GDK), see the <i>Verity Gateway Developer's Kit Programming Reference</i> .
<code>-help</code>	CRWL CTRL INDX	Prints this help.
<code>-https</code>	CRWL ---- ----	Enables access to SSL-secured Web sites (must install the SSL option pack).
<code>-indexer c_host:port</code>	---- ---- INDX	Starts an Indexer component and registers it under the specified Controller.

Usage	Components			Description
-indxsessions <i>num</i>	----	----	INDX	Specifies the number of concurrent indexing sessions [2].
-ip <i>ipaddress</i>	CRWL	CTRL	INDX	Binds k2spider_srv to the specified IP address.
-keyfile <i>filepath</i>	CRWL	----	----	Specifies the Private Key file to use with -certfile.
-license <i>path</i>	CRWL	CTRL	INDX	Specifies the K2 Spider license file.
-locale <i>name</i>	----	----	INDX	Specifies the collection's locale.
-loglevel <i>level</i>	CRWL	CTRL	INDX	Specifies the logging level for a component. Choose from one of the following: skip summary verbose debug trace
-maxindfilehandles <i>N</i>	----	----	INDX	Specifies the maximum number of file handles per VDK session for an Indexer (-indexer).
-maxindmem <i>k</i>	----	----	INDX	Specifies in kilobytes the maximum amount of memory for an Indexer (-indexer).
-msgdb <i>filepath</i>	CRWL	CTRL	INDX	Specifies the path to the message database.
-noindex	----	----	INDX	Specifies (BIFs) are submitted but not yet indexed.
-nooptimize	----	----	INDX	Specifies no optimization for the collection.
-numpages	----	----	INDX	Sets VDK session disk cache size (pages).
-output	CRWL	CTRL	INDX	Specifies file to which stdout and stderr are redirected.
-port <i>num</i>	CRWL	CTRL	INDX	Specifies the port number for components of this service instance.
-recover	----	CTRL	----	Specifies the Controller component is to run in recover mode.
-style <i>dirpath</i>	CRWL	----	INDX	Specifies the path to the collection style path.
-submitsize <i>num</i>	----	----	INDX	Specifies the number of documents to be processed at a time [1024].
-temp <i>dirpath</i>	CRWL	----	INDX	Specifies the path to the temp directory for the job.
-vdkclean <i>N</i>	----	----	INDX	Cleans up the VDK session after indexing every <i>N</i> documents.
-workpath <i>dirpath</i>	CRWL	CTRL	INDX	Specifies the path for all of the working files and directories for the k2spider_srv component.

Using K2 Spider Client

This chapter describes how to use the K2 Spider Client.

Note You cannot use different versions of K2 Spider Server (`k2spider_srv`) and K2 Spider Client (`k2spider_cli`). K2 Spider Server and K2 Spider Client must be the same version.

This chapter includes the following sections:

- Starting the K2 Spider Client
- Using `k2spider_cli` to Index
- Managing `k2spider_srv` Services
- Using the Bulk Submit Feature
- Deleting Keys and Records with VSDB
- Reporting
- `k2spider_cli` Reference

Starting the K2 Spider Client

The first step to managing the K2 Spider system is to initiate a K2 Spider Client session and connect to the Controller. You use the `k2spider_cli` service to interact with the rest of the K2 Spider system.

To connect to a Controller, enter the following command:

```
% k2spider_cli -controller host_or_IP -port portnum
```

where *host_or_IP* is the host name or IP address on which the `k2spider_srv` service is running as a Controller, and *portnum* is the port number you specified for the Controller component of the `k2spider_srv` service.

When you press enter to execute the command, you enter a K2 Spider Client session. You will see output similar to the following:

```
% k2spider_cli -controller caliber -port 2222
k2spider_cli - Verity, Inc. Version 6.0.0 (_nti40, Oct 25 2003)

*****Connected to hostName=CALIBER portNum=2222*****
**Remote Server Info=192.168.0.11:2222@K2Spider_Server$6.0.0**

Type '?' for a list of commands

k2spider_cli>
```

In this instance, CALIBER is the host on which the `k2spider_srv` service is running, and 2222 is the port number for the Controller component of that service.

Now that you are in a K2 Spider Client session, you can manage and interact with the K2 Spider system.

Migrating K2 Spider Jobs

For information on using the `-migrate` and `-migratevsdb` options for K2 Spider Client, see the *Verity K2 Migration Guide*.

Options for Using k2spider_srv as a Controller

The supported options for the k2spider_srv service when it is run as a Controller are described below. The **Type** indicator is used to specify unique usage requirements. For example, the entry may indicate that the option is required or that the option is used only with a particular gateway type.

batch

Description The batch command enables you to group commands together in a text file.

Syntax `batch batch_file`

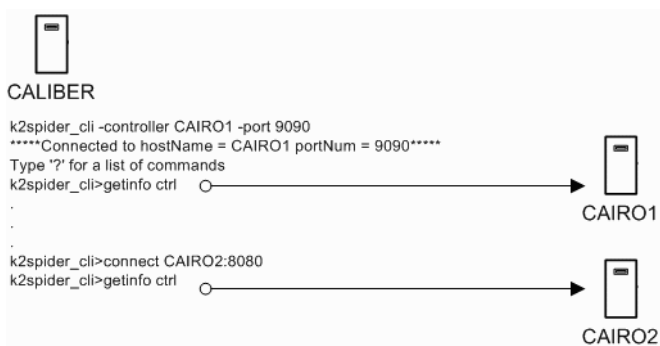
connect

Description The connect command enables K2 Spider Client to connect to a Controller other than the one to which you connected when you initiated the K2 Spider Client session.

Syntax `connect controllerhostname:controllerportnum`

Details When you connect to another Controller in a K2 Spider Client session, you are automatically disconnected from the current Controller. You can be connected to only one Controller at a time.

Figure 8-1 K2 Spider Client session



create

Description The `create` command creates the job specified by *inifilepath* on the Controller to which you are currently connected.

Syntax `create inifilepath`

Details Once the job is created on a Controller, it will run according to its schedule mode as specified in the job INI file, or when you next issue a `start` command.

Note Although you can specify the root name of the file, the actual file must have an extension of `.ini`. For example, for a job file named `myjob1.ini`, you can specify `myjob1` for *inifilepath*.

See Also For more information, see [“Creating Indexing Jobs” on page 222](#).

createcoll

Description The `createcoll` command creates the directory structure for an empty collection without creating the indexing job.

Syntax `createcoll crwl|indx crwl_or_indx_spec collname gatewaytype styledir [authfilepath]`

- `crwl|indx` indicates the type of component to run
- `crwl_or_indx_spec` is the host name and port number of the component (`crwl` or `indx`) you specified as the type of component

The component must be running when you issue the `createcoll` command.

- `collname` is a collection name in the server’s work path or the absolute path to a collection

- `gatewaytype` specifies that the gateway is one of the following types:

`filesystem|web|documentum|exchange|notes|odbc`

- `styledir` is a directory on the local machine that runs the client that contains valid style files

When specifying *styledir*, you must use the style files appropriate for each indexing gateway. Style files created with the K2 Dashboard are located in:

`dataDir/stylesets/stylesetname`

where *dataDir* is the data directory. For information on the K2 Dashboard and creating style sets, see the *Verity K2 Dashboard Administrator Guide* and the relevant gateway guide if necessary.

- *authfilepath* is the required authorization file that is used when the gateway requires password authentication. Its format must be the same as the K2 Spider auth file. It is located on the host on which runs the K2 Spider Client.

Details You can start Indexers and Crawlers without specifying a collection and style files. At a later time, use the `createcoll` command to create the collection.

Examples In these examples, the command must be entered as a single, continuous string at the command-line. The command is broken up here due to margin constraints.

Before you issue the `createcoll` commands contained in these examples, make sure the associated Crawlers and Indexers identified in the commands are running.

Example 1:

```
createcoll indx caliber:9800 C:\Ocolls\vrtpage filesys  
C:\Program Files\Verity\Data\stylesets\def_filesystem
```

This example uses an Indexer at port 9800 on the host `caliber`. The absolute path to the collection is `C:\Ocolls\vrtpage`. The File System gateway `filesys` uses the style files in the specified path.

Example 2:

```
createcoll crwl caliber:9800 notes_coll notes  
C:\Program Files\Verity\Data\stylesets\notes_style
```

This example uses a Crawler at port 9800 on the host `caliber`. The collection name is `notes_coll`. The Notes gateway `notes` uses the style files in the specified path.

Example 3:

```
createcoll indx caliber:9800 sec_coll web  
C:\Program Files\Verity\Data\stylesets\def_http_secure  
C:\Program Files\Verity\Data\sec_coll\style\my_auth_file
```

This example uses an Indexer at port 9800 on the host `caliber`. The collection, named `sec_coll`, has a gateway type of `web` and contains secure documents. A style file and an *authfilepath* are required.

Note When attempting to index Notes, Exchange, ODBC, or Documentum repositories, you must supply the style files on the Crawler side so that it can configure the gateway. There are two times when this can be done:

- At Server start-up by specifying the `-style gateway_style -gateway gateway_type` options
- Using the `createcoll` command to create a pseudo collection on the Crawler by using the corresponding style files and gateway type

The Crawler will create a pseudo collection with a temporary predetermined name in the workpath. However, the collection name specified will eventually be used.

If you intend to use the same `k2spider_srv` service running a Crawler component for multiple sessions where different gateways are used, you should delete the contents of the workpath (as specified by `-workpath`) between each session.

delete

Description The `delete` command deletes the job specified by *jobname* on the Controller to which you are currently connected.

Syntax `delete jobname`

Details In addition to deleting a job from the Controller's memory, the `delete` command also deletes the job INI file, all related databases, and other files from the file system. Once a job has been deleted from a Controller, you must use the `create` command to create it again before you can run it.

This command only removes remnants of a job outside of a collection. The document keys indexed into a collection (by the job you want to delete) are not affected.

Note You must first use the `stop` command to stop the job you intend to delete.

See Also For more information, see [“Deleting an Indexing Job” on page 225](#).

encrypt

Description The `encrypt` command enables you to create an encrypted version of a value for use in the job INI files themselves and files referenced by the `auth` option.

Syntax `encrypt value`

getinfo

Description The `getinfo` command reports all known services, depending on the service specified, along with pertinent statistics such as number of document keys and the current state.

Syntax `getinfo ctrl|crl|indx`

See Also For more information, see [“Using getinfo” on page 236](#).

getstat

Description The `getstat` command reports the current state of the specified job along with indexing statistics.

Syntax `getstat jobname`

Details The possible states for a job are: Running, Finished, Stopping, Stopped, Paused, Scheduled, Purging, and Deleting.

Note Statistics are re-set for each job each time the job is run.

See Also For more information, see [“Using getstat” on page 238](#).

getjobini

Description The `getjobini` command enables K2 Spider Client to get a job’s INI file based on the job name and copy it to a new location.

Syntax `getjobini jobname inifilename`

Details Issue this command to use existing jobs as templates to create new jobs when you do not remember where a particular job’s INI file is located.

Note The INI file retrieved by `getjobini` may not be the same as the original. The Controller changes paths to the original `auth`, `metamap`, `mimemap` and `prefixmap` files to copies which are local to the Controller. If these files are specified in the job INI file, you will need to edit the paths to refer to the proper files relative to the K2 Spider Client before re-submitting the job to the Controller.

Although you can specify the root name of the job initialization file, the actual file must have an extension of `.ini`. For example, for a job file named `myjob1.ini`, you can specify `myjob1` for *jobname*.

help

Description The `help` command prints the available K2 Spider Client commands to standard output.

Syntax `help [command_name] or ? [command_name]`

Note Optionally specify a command name to get help on that specific command.

modify

Description The `modify` command enables K2 Spider Client to edit a job's INI file while it is managed by a Controller.

Syntax `modify inifilename`

Details The alternative to using the `modify` command is to first delete the job on the Controller, then modify the local copy of the INI file, then create the job on the Controller again. The `modify` command enables the job to remain on the Controller.

Note If the job whose INI file you want to modify is running, you must first stop that job using the `stop` command. Use the `getinfo` or `getstat` command to find out if your jobs are still running.

Although you can specify the root name of the job initialization file, the actual file must have an extension of `.ini`. For example, for a job file named `myjob1.ini`, you can specify `myjob1` for *inifilename*.

If the job you are modifying is not scheduled, remember to manually start it again once you are done modifying it. Otherwise, the job will remain stopped.

pause

Description The `pause` command temporarily halts indexing activity for the specified *jobname*.

Syntax `pause jobname`

Note Use `resume` to begin indexing again on any paused jobs.

purge

Description The `purge` command enables K2 Spider Client to mark for deletion all keys in the job's persistent store.

Syntax `purge jobname`

Details The persistent store of each job is kept synchronized with the collection. You can only purge one job's keys at a time. When the Controller receives the `purge` command from the user, it performs the following steps:

1. The Controller changes the job status to `Purging`.
2. The Controller searches the persistent store for keys that were indexed into the collection by the specified *jobname*.
3. The Controller submits those keys to corresponding Indexers for deletion.
4. The Indexers immediately acts and delete the keys.

Note The Indexers must be running or you will see an error message that the keys could not be deleted. If an Indexer is down, only the keys in the appropriate persistent store are removed. The keys are not removed from the relevant collection. You must manually delete those keys from the collection.

5. The Indexers then notify the Controller that the keys have been deleted from the collection.
6. The Controller changes the job status to `Finished` and prints out a summary message after it verifies that the Indexer has finished.

Note If the job whose keys you want to purge is running, you must first stop that job using the `stop` command.

quit

Description The `quit` command terminates the K2 Spider Client session.

Syntax `quit` or `q`

resume

Description The `resume` command resumes indexing activity for the specified paused *jobname*.

Syntax `resume jobname`

Note Use `resume` only for paused jobs.

start

Description The `start` command runs the specified job which must already have been created on the Controller.

Syntax `start jobname`

Details Use the `start` command to run a scheduled job at a time other than its scheduled time or to run an unscheduled job.

Note You can use the `start` command for jobs whose status is Inactive.

stop

Description The `stop` command causes all indexing activity to halt for the specified *jobname* once all queued keys are processed.

Syntax `stop jobname`

Note Jobs which you intend to `purge`, `modify`, or `delete` must be stopped first.

shutdown

Description The shutdown command causes the specified service to exit.

Syntax shutdown *ctrl* | *serverspec*

Details If you specify shutdown *ctrl*, all Crawlers and Indexers managed by the Controller to which you are connected will be shutdown. If you want to shut down only specific services, then you must use shutdown *serverspec* where *serverspec* is the host name and port number of the particular service.

For example, if you want to shut down an Indexer at port 9090 on host CAIRO, then use:

```
K2S Client>shutdown CAIRO:9090
```

Note Before you shut down a service, ensure that all running jobs controlled by that service are stopped by using the [stop](#) command. If any jobs are running when you attempt to shut down a service, you will see a message reminding you to stop the jobs.

submit bif

Description The submit bif command enables K2 Spider Client to submit a standard Verity Bulk Insert File (BIF) for indexing by the specified job.

Syntax submit bif *submfile* *jobname* *indxspec* *submitsize*
insert | update | modify | delete

Details The specified job must be in either a Running or a Finished state.

See Also For more information, see [“Using the Bulk Submit Feature” on page 227](#).

submit doc

Description When the Indexer cannot access the documents, the `submit doc` command enables K2 Spider Client to submit an individual document for indexing by the specified job.

Syntax `submit doc document jobname [indxspec]`

Details The specified job must be in either a Running or a Finished state. For example:

```
% k2spider_cli>submit doc /docs/a123.txt myjob cairo:9191
```

where `myjob` is the name of the job which must be either Running or Finished, `cairo` is the host name of the machine on which the `k2spider_srv` service is running, and `9191` is the port number for the Indexer component of the `k2spider_srv` service. The document must be accessible to the computer on which you are running the K2 Spider Client.

Warning! The `submit doc` command copies the document to the `temp` directory in the workpath of the Indexer being used. This path, relative to the collection into which the document is being indexed, and a randomly generated name are stored in the `VdkVgwKey` field for the document. Keep in mind that this path becomes the source for the documents for viewing, so you cannot clear the workpath for the Indexer. Doing so would break any future attempts by users to view the documents.

As this may not be desirable, you should, if at all possible, copy or move the documents so that Indexers have access to them and you can then use the `submit bif` or `submit keys` command to submit the documents for indexing.

submit keys

Description The `submit keys` command enables K2 Spider Client to submit a text file containing a list of files for indexing by the specified job.

Syntax `submit keys filelist jobname [indxspec]`

Details The specified job must be in either a Running or a Finished state. For example:

```
% k2spider_cli>submit keys job143list myjob CAIRO:9191
```

where `job143list` is the name of the text file which contains a list of files, `myjob` is the name of the job which must be either Running or Finished, `CAIRO` is the host name of the machine on which the `k2spider_srv` service is running, and `9191` is the port number for the Indexer component of the `k2spider_srv` service.

The syntax for the file list is `path/filename` with each entry on a separate line.

VSDB

Description The VSDB command creates a special session with a collection's persistent store. Within that session, you can use VSDB options to get information from the persistent store.

Syntax VSDB

Note The VSDB session within K2 Spider Client is not the same as running the vsdb command-line tool (for Verity Spider) found in the /bin directory. Persistent stores created by Verity Spider are not compatible with persistent stores created by K2 Spider.

See Also For more information, see [“Using VSDB” on page 240](#).

Using k2spider_cli to Index

To run an indexing job, use the K2 Spider Client to create the job on the Controller. Once the job has been created with the Controller, it will run as scheduled if you specified a schedule mode in the job INI file, or you can use the K2 Spider Client, or k2spider_cli service, to start the job at any time.

Creating Indexing Jobs

To create the indexing job on the Controller, do the following.

1. Start the k2spider_cli service as described in “Starting the K2 Spider Client” earlier in this chapter.
2. Create the indexing job by entering the following command.

```
% k2spider_cli>create path/job.ini
```

where *path/job.ini* is the file name of your job INI file. For information on the job INI file, see [Chapter 9, “Job Initialization File Reference.”](#)

Note that the root of the file name will become the indexing job’s name throughout the K2 Spider system for the duration of the job’s existence on the Controller. For example, you will need to refer to the job name when you perform actions such as *start*, *stop*, *resume*, *delete*, *getstat*, *abort*, or *submit*.

At this point, the indexing job is known to the Controller. If you have specified a schedule mode, then the indexing job will run without any further intervention.

Managing Indexing Jobs

Once an indexing job has been created on the Controller, you can manage it by using the available `k2spider_cli` options. For more information, see [“k2spider_cli Reference” on page 246](#).

Management options include the following.

- [Starting an Indexing Job](#)
- [Stopping an Indexing Job](#)
- [Modifying an Indexing Job](#)
- [Pausing an Indexing Job](#)
- [Resuming an Indexing Job](#)
- [Deleting an Indexing Job](#)

Starting an Indexing Job

If you did not specify a schedule in the job INI file, or if you want to run an indexing job at a time other than the scheduled time, then you will need to manually start the indexing job.

From the K2 Spider Client prompt, enter the following command to start an indexing job for which there is an INI file named `jobname.ini`:

```
% k2spider_cli>start jobname
```

Stopping an Indexing Job

In order to delete, modify or purge a job, you must first stop it.

From the K2 Spider Client prompt, enter the following command to stop an indexing job for which there is an INI file named `jobname.ini`:

```
% k2spider_cli>stop jobname
```

The job will change states from Running to Stopping to Stopped. It may take some time until it is Stopped. You can use the `getstat` option to check on the state of a job. For more information, see [“Using getstat” on page 238](#).

Once the job has stopped, you can [purge](#), [modify](#), or [delete](#) it.

Modifying an Indexing Job

Modifying a job involves making changes to the job's INI file while the job is created on the Controller. Rather than deleting the job from the Controller and then creating it again, you can modify the job.

To modify a job, do the following.

1. Open a job INI file in a text editor and make your changes.
2. Start a K2 Spider Client session as described in "Starting the K2 Spider Client."
3. Stop the job for which you are modifying the INI file. For example:

```
% k2spider_cli>stop jobnumber1
```

You must first stop a job before you can modify it.

4. Enter the following command at the K2 Spider Client prompt.

```
% k2spider_cli>modify path/jobnumber1.ini
```

Unlike with `stop`, where you just use the root name of the job INI file, with `modify` you must specify the full path and name for the job INI file. The file name in this example includes the extension `.ini`. Full job file names can be any valid file name.

Pausing an Indexing Job

If for some reason you want to suspend work on an indexing job without actually removing the job from the working K2 Spider system, you can pause the indexing job and resume it at a later time.

From the K2 Spider Client prompt, enter the following command to pause an indexing job for which there is an INI file named `jobname.ini`:

```
% k2spider_cli>pause jobname
```

Use the `resume` command to resume the indexing job.

Resuming an Indexing Job

From the K2 Spider Client prompt, enter the following command to resume an indexing job for which there is an INI file named `jobname.ini`:

```
% k2spider_cli>resume jobname
```


Deleting an Indexing Job

When you no longer want an indexing job to exist in the file system or reside in the Controller's memory, you can delete it.

To delete an indexing job, do the following.

1. Start the `k2spider_cli` service as described in ["Starting the K2 Spider Client" on page 210](#).

2. Stop the indexing job by entering the following command.

```
% k2spider_cli>stop jobname
```

3. Delete the indexing job by entering the following command.

```
% k2spider_cli>delete jobname
```

Once the indexing job has been deleted from the Controller and the file system, you must create it again before you can run it. For more information, see ["Creating Indexing Jobs" on page 222](#).

Managing k2spider_srv Services

With the K2 Spider Client, you can remotely shutdown k2spider_srv services. When k2spider_srv services crash, you can recover them so they will resume their work.

Shutting Down Services

To shut down a k2spider_srv service, do the following.

1. Start the k2spider_cli service as described in [“Starting the K2 Spider Client” on page 210](#).
2. Stop all jobs associated with that service using the K2 Spider Client [stop](#) option.

```
% k2spider_cli>stop job1
% k2spider_cli>stop job2
% k2spider_cli>stop job3
% k2spider_cli>
```

3. Use the [getinfo](#) option to ensure that all of the jobs are stopped.

```
% k2spider_cli>getinfo ctrl
```

4. Shut down the desired service using the K2 Spider Client [shutdown](#) option.

```
% k2spider_cli>shutdown service
```

where *service* is either *ctrl* for the Controller, or a *serverspec* for a particular k2spider_srv service such as a Indexer. For example, to shut down a Indexer at port 9102 on CALIBER, use the following command.

```
% k2spider_cli>shutdown CALIBER:9102
```

If you did not stop all running jobs for the service you are shutting down, you will see a message reminding you to stop them.

Using the Bulk Submit Feature

The bulk submit feature enables K2 Spider Client to:

- Insert field values for one or more documents in a collection
- Insert numerous documents into a collection

Connecting to a Controller

The first step is to connect to the Controller. For more information, see [“Starting the K2 Spider Client” on page 210](#). Once you have established a K2 Spider Client connection to the Controller, you can use the `submit bif` command as follows.

1. Define the desired document-related fields in the appropriate style file.
2. Create a bulk submit file to specify the documents to submit.
3. Use `submit bif` to submit the documents.

Defining Document-Related Fields

To define document-related fields, you must include the field definitions in the `style.ufl` file and/or the `style.sfl` file. For information about defining fields for the collection schema using these files, refer to the “Field Definitions” chapter in the *Verity Collection Reference Guide*.

The `VdkVgwKey` field specifies the document key and it is required. The `VdkVgwKey` field is defined in the default `style.ddd` file shipped with every Verity product (the default `style.ddd` file should not be edited).

Note `VdkVgwKey` values must be unique across all collections that will be searched in any one application to avoid having the `VdkVgwKey` submitted and found multiple times.

Creating a Bulk Insert File (BIF)

A bulk insert file is used to specify the documents to be submitted to a collection. For each document, the following information can be supplied: field names and values, document size (in bytes), and document location in the file system.

Every line in a bulk insert file is a statement. Bulk insert files contain the following types of statements:

- Comments
- Field definitions
- Document record delimiters

Comments

Blank lines and lines that begin with a pound sign (#) are interpreted as comments.

Field Definitions

Fields are defined using the following syntax rules:

- Field names must begin in the first column of the bulk submit file.
- Field names are not case sensitive.
- Field names can contain any alphanumeric character, as well as the hyphen (-) and underscore (_).
- Field names must be terminated with a colon (:).
- Field values begin after the colon and continue to the end of the line.
- Leading and trailing blank spaces are stripped from the field value when the bulk submit file is read.
- Single and double quotes surrounding the field value are stripped when the bulk submit file is read. Blank spaces that occurred between the quotes are preserved.
- C-style escape sequences (such as \t representing a tab) are allowed in field names.
- A group of `field:value` lines is a document record delimited by `<<EOD>>`. See [“Example 1” on page 231](#).

End-of-Document Indicators

Ends of documents are indicated using the following statement:

```
<<EOD>>
```

Control File Parsing

Pathname Interpretation

The Verity search engine interprets UNIX style syntax on all platforms, as well as platform-specific syntax. Note that if you are running on Windows, you must specify two backslash characters in pathnames instead of one. For example, a Windows pathname such as this:

```
C:\markets\data
```

must be specified as:

```
C:\\markets\\data
```

Escape Sequences

The lead-in character for an escape sequence is the backslash. An escape sequence is a set of characters that, together, represent one character that may have a special significance. An escape sequence may be embedded within a character constant or string literal. In a string literal, the apostrophe may be represented by itself or by its escape sequence. In a character constant, the quotation mark may be represented by itself or by its escape sequence.

Character Set of Bulk Insert File (BIF)

The bulk insert file (BIF) should always be in the character set for the current locale for the collection. For information on supported locales and their use, see the *Verity Locale Configuration Guide*.

Characters having special meanings are used to specify non-printable ASCII characters or characters special to the Controller such as backslash and quotes, as indicated below.

Table 8-1 Non-printable ASCII characters

This string:	Is interpreted as:
<code>\b</code>	backspace
<code>\e</code>	escape
<code>\f</code>	formfeed
<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\t</code>	tab
<code>\v</code>	vertical tab
<code>\xnn</code>	a character represented by the hexadecimal value nn
<code>\\</code>	backslash
<code>"</code>	double quote
<code>'</code>	single quote

Defining Backslash as a Literal Character

If you want a backslash to be interpreted as a literal character, the BIF syntax must include two backslashes instead of one backslash. When the Verity search engine encounters two backslashes in a row, it strips the first one and interprets the second as a literal backslash.

Example 1

The following sample file shows the Bulk Insert File syntax.

```
# Every document must have a key
VdkVgwKey: 538592765
# Construct the pieces of the dispatch field that will
# point at the text. The "document" starts at byte offset 1400
# in text/d940116.txt and extends for 945 bytes.
DOC_FN: text/d940116.txt
DOC_OF: 1400
DOC_SZ: 945
# Embed a hex character into a field value
Company: F.SNM \x1a Ti Group Plc (U.TI)
# Use quotes to preserve leading or trailing white space
Hot: " "
# End the document, then start a new one
<<EOD>>
# The second document
VdkVgwKey: 538592766
DOC_FN: text/d940116.txt
DOC_OF: 2345
DOC_SZ: 690
Ddate: 16-Jan-94 06:05 am
Headline: *Zurich Noon Gold At 1100 GMT
Hot: "H"
<<EOD>>
```

Example 2

The following sample Bulk Insert File contains four documents. The `TITLE` and `ACTOR` fields will be populated if they are defined in the collection's `style.ufl` file.

```
TITLE:          Dances with Wolves
ACTOR:          Kevin Costner
VDKVGWKEY:      docs/articl26
<<EOD>>
```

```
TITLE:          Ghost
ACTOR:          Patrick Swayze, Demi Moore, Whoopi Goldberg
VDKVGWKEY:      docs/articl28
<<EOD>>
```

```
TITLE:          Nightmare Before Christmas
VDKVGWKEY:      docs/articl29
<<EOD>>
```

```
TITLE:          Gone with the Wind
ACTOR:          Vivien Leigh, Clark Gable
VDKVGWKEY:      docs/articl27
<<EOD>>
```


Submitting the Bulk Insert File

In the K2 Spider Client session, enter the following command.

```
% k2spider_cli>submit bif bif_name myjob cairo:9191  
    submitsize 10 insert
```

where *bif_name* is the name of the BIF, *myjob* is the name of the job which must be either Running or Finished, *cairo* is the host name of the machine on which the *k2spider_srv* service is running, 9191 is the port number for the Indexer component of the *k2spider_srv* service, *submitsize* is a required option for specifying a reasonable divisor for the workload and *insert* is the action to be taken with the information in the BIF.

Note The value for *submitsize* should reasonably break up the total number of document keys you are submitting in the BIF. For example, if the BIF contains 1000 document keys, you might want to make 200 the value of *submitsize*. If the BIF contains a small number of document keys, then make the total number the value of *submitsize*. Keep in mind that *submitsize* is required and you must supply a number that is less than or equal to the total number of document keys in the BIF.

The contents of the BIF are kept on the Controller for the duration of the session. For this reason, you should keep the number of document keys in the BIF to a manageable number that does not consume too much memory. A rule of thumb is to keep the total number to around 1000 for optimum performance.

Submit Actions

There are several actions to choose from when using the `submit bif` command.

- Use `insert` when you are first submitting a BIF to an indexer.
- Use `update` when you are adding new document keys to a collection.
- Use `modify` when you have changed meta data information in a BIF that has already been submitted to an indexer and is again being submitted to the same indexer.
- Use `delete` when you want to remove document keys from a collection.

Syntax `submit bif submfile jobname indxspec submitsize
insert | update | modify | delete`

Deleting Keys and Records with VSDB

Using a VSDB session, you can delete records from only a job's persistent store, only the collection, or you can delete them from both the persistent store and the collection.

Using -delete

With the VSDB `-delete` option you can mark records for deletion in the persistent store. The document keys will remain in the collection.

To delete records from just a persistent store:

1. Within a K2 Spider Client session, start a VSDB session by entering the following command.

```
% k2spider_cli>VSDB
```

The prompt will change to indicate you are now in a VSDB session. For more information on VSDB, see ["Using VSDB" on page 240](#).

```
% k2spider_cli\VSDB>
```

2. Create a new VSDB object and execute the `delete` command by entering the following command.

```
% k2spider_cli\VSDB>new jobname -delete criteria flag
```

where *jobname* is the name of the job for which you want to mark records for deletion and *criteria* is `-match`, `-key` or `-status`.

For more information, see ["-delete" on page 243](#).

3. To also remove the matching document keys from the collection, use the same *expression* value with `-delcoll` as described in ["Using -delcoll" on page 235](#).

Using -delcoll

Using the `-delcoll` option deletes document keys matching your criteria from the collection, leaving those document keys in the persistent store. You may want to do this to ensure the next run of the indexing job gets all of the latest documents.

To only remove the matching document keys from the persistent store, use `-delete`.

1. Within a K2 Spider Client session, start a VSDB session by entering:

```
% k2spider_cli>VSDB
```

The prompt will change to indicate that you are now in a VSDB session.

```
% k2spider_cli\VSDB>
```

2. Create a new VSDB object and execute the `-delcoll` command by entering the following command.

```
% k2spider_cli\VSDB>new jobname -delcoll criteria options
```

where *jobname* is the name of the job for which you want to mark records for deletion, and *criteria* is one of `-match`, `-key` or `-status`, and *options* are any of `-showrec`, `-regex` and `-casesen`. For example:

```
% k2spider_cli\VSDB>new powerpoint -delcoll -match *.ppt
```

For more information, see [“VSDB Options” on page 243](#).

Purging Document Keys and Records

Rather than individually and selectively marking for deletion records from a job’s persistent store and document keys from a collection, you can purge a job’s persistent store and all of that job’s document keys from the collection with the K2 Spider Client `purge` command.

To purge a job’s persistent store and all of that job’s document keys from the collection, enter the following command from within a K2 Spider Client session.

```
% k2spider_cli>purge jobname
```

where *jobname* is the name of the job for which you want to purge all document keys from both the persistent store and the collection.

Note The Indexer serving the job must be running or you will see an error message regarding the keys not being deleted from the collection.

Reporting

With the K2 Spider Client reporting options, you can gather information about the status of various aspects of the K2 Spider system.

Connecting to a Controller

The first step in gathering information about what is happening in your K2 Spider system is to connect to the Controller. For more information, see [“Starting the K2 Spider Client” on page 210](#).

Once you have established a K2 Spider Client connection to the Controller, you can use one of three K2 Spider Client options to gather information about such items as the status of various services or the statistics of a given indexing job.

The K2 Spider Client options available for information gathering are:

- `getinfo`
- `getstat`
- `VSDB`

Using `getinfo`

The K2 Spider Client `getinfo` option enables you to view a report about everything the specified service knows, which may include current indexing jobs, Crawlers, Indexers and their statistics.

To use the `getinfo` option, do the following.

1. Start the `k2spider_cli` service as described in [“Starting the K2 Spider Client” on page 210](#).
2. Get the appropriate information by entering the following command.

```
% k2spider_cli>getinfo service
```

where *service* is one of `ctrl` for Controller, `crwl` for Crawler or `indx` for Indexer.

Note When the Crawler/Indexer is started without specifying the gateway type or collection attached, the default datatype is UNKNOWN. Although the datatype is UNKNOWN, the Crawler handles both the FILESYS and WEB datatypes as if they were the defaults. Once a job has been created, the appropriate datatype appears in the output.

Sample Output for getinfo ctrl Command

```
Job Name = myJob
Job State = Running
Job Name = yourJob
Job State = Finished
Total job count = 2

Controller hostName = caliber.company.com
Controller hostPort = 9090

Crawler(s) connected to this controller
Crawler 0 hostName = cairol.company.com
Crawler 0 hostPort = 9191
Crawler 0 keyCount = 1324
Crawler 0 dataType = FileSys and Web
Crawler 0 state = running

Indexer(s) connected to this controller
Indexer 0 hostName = cairol.company.com
Indexer 0 hostPort = 9192
Indexer 0 keyCount = 1520
Indexer dataType = Web
Indexer 0 collPath = c:\coll1
Indexer 0 collName = coll1
Indexer 0 state = running
```

Sample Output for getinfo crwl Command

```
Crawler(s) connected to this controller
Crawler 0 hostName = cairol.company.com
Crawler 0 hostPort = 9191
Crawler 0 keyCount = 1324
Crawler 0 dataType = FileSys and Web
Crawler 0 state = running
```

Sample Output for getinfo indx Command

```
Indexer(s) connected to this controller
Indexer 0 hostName = cairol.company.com
Indexer 0 hostPort = 9192
Indexer 0 keyCount = 1520
Indexer 0 dataType = Web
Indexer 0 collPath = c:\coll1
Indexer 0 collName = coll1
Indexer 0 state = running
k2spider_cli>
```

Using getstat

The K2 Spider Client `getstat` option enables you to view a report of a job's indexing statistics.

1. Start the `k2spider_cli` service as described in "Starting the K2 Spider Client" earlier in this chapter.
2. Get the appropriate information by entering the following command.

```
% k2spider_cli>getstat jobname
```

Sample Output for getstat jobname Command

The following sample output is for an indexing job named `myjob`.

```
k2spider_cli>getstat myjob
Job Name = myjob
Job State = Running
Keys to be dispatched to Crawlers = 0
Keys dispatched to Crawlers = 8
Failed to fetch keys = 8
Crawled keys = 390
Skipped keys = 212
Keys dispatched to Indexers = 178
Indexed keys = 0
Deleted keys = 0
Bad keys = 0
Crawlers in sleep mode = false
Job status on Crawler: cairol.company.com:9192
  Keys to be crawled = 2
  Keys to be submitted to Controller = 0
```

```
Job status on Indexer: cairol.company.com:9192
  Keys to be indexed or updated = 50
  Keys to be deleted = 0
  Number of BIF files = 1
  Keys to be downloaded from Crawlers = 0
  Keys to be submitted to Controller = 0
  Indexer (session#0) is working on key =
    http://www.yourcompany.com/path/filename.htm
Execution time: 0 minutes 50 seconds
k2spider_cli>
```

Following is sample output for the same getstat command run again.

```
k2spider_cli>getstat test1
Job Name = myjob
Job State = Finished
Failed to fetch keys = 8
Crawled keys = 392
Skipped keys = 212
Indexed keys = 171
Deleted keys = 0
Bad keys = 9
Execution time: 2 minutes 57 seconds
k2spider_cli>
```

Using VSDB

During an indexing job, document keys can be in any one of several processing stages. With the VSDB option, you can generate a report about the status of document keys in a job's persistent store.

To enter into a VSDB session, do the following.

1. Within a K2 Spider Client session, start a VSDB session using the VSDB option. Note that the prompt will change to indicate you are in a VSDB session.

```
% k2spider_cli\VSDB>
```

2. Establish an object with an indexing job by entering the following command.

```
% k2spider_cli\VSDB>new jobname options
```

where *jobname* is the name of the job for which you want to gather information from the persistent store, and *options* are any of the available VSDB options as described in the next section.

For example, to view the persistent store records for document keys which match the expression *'*.html'*, use:

```
% k2spider_cli\VSDB>new job1 -showrec -match *.html
```

Another example queries for records of failed document keys using:

```
% k2spider_cli\VSDB>new job1 -showrec -status fail
```

3. Use the next action to view the next set of records. You can affect the number of records returned with the next action by adding *-keycount num_records* to the new action.

```
% k2spider_cli\VSDB>new job1 -keycount 30 -showrec -status fail  
% k2spider_cli\VSDB>next
```

Instead of the default ten records, you will see thirty records.

4. When you are finished with a particular job, you can create a new object with a different indexing job, you can free the job object and leave the VSDB session open, or you can quit the VSDB session.

Sample VSDB Session

For this example, a small directory structure was indexed from a Windows 2000 file system. For information on the actions and options available in a VSDB session, see [“VSDB Actions” on page 242](#) and [“VSDB Options” on page 243](#).

Following are the parameters for the example:

- The indexing job INI file is `getfiles.ini` which makes the job name `getfiles`.
- Including the starting point, there are ten total directories.
- There are a total of two-hundred and twenty-three files to be indexed.

Use the VSDB option to find out the following:

- How many files were indexed?
- Were any files skipped, and if so, which files?

Here is the K2 Spider Client sample session using VSDB:

```
C:\verity\_nti40\bin\k2spider_cli -controller cairo -port 9000
K2Spider_Client-Verity, Inc. Version 6.0.0 (_nti40, July 15,
2005)
****Connected to hostName = lucid portNum = 9000****
Type '?' for a list of commands

k2spider_cli>VSDB
k2spider_cli\VSDB>new getfiles -status done
Summary: 0 records deleted from database.
Summary: 0 keys deleted from collection.
Summary: 223 total, 211 in collection, 212 matching query
1 none
k2spider_cli\VSDB>new getfiles -status none -showrec
1 matching record
1 C:\data\dir1\subdir1\skippedfile.wri C
Summary: 0 records deleted from database.
Summary: 0 keys deleted from collection.
Summary: 223 records total, 211 keys in collection, 1 record
matching query
```

As you can see from the output, 212 files were processed (status done) with 1 file skipped (status none) and 211 files actually indexed into the collection. The 10 records not indexed (10 cand) are the directories. Adding `-showrec` to the query for records with a status of none displays the name of the skipped file. Although directories are included in counts reported by VSDB, they are not actually keys indexed in the collection.

VSDB Summaries

When VSDB reports summaries, persistent store records are counted for the total, while document keys are counted for the value in the collection. Directories are stored in the persistent store and count as records, but they do not exist in the collection and do not count toward the `in_collection` value.

For example, if you indexed a directory which contained three subdirectories and among all directories were 216 documents, you might see the following summary line.

```
Summary: 220 records total, 216 keys in collection, 110 records  
matching query
```

In this example, the four directories are counted in the total because they exist as persistent store records. The total count for `in_collection` is four less because only document keys exist in the collection. The value for `matching_query` depends on your query.

VSDB Actions

The following table contains the supported actions for a VSDB session.

Action	Description
<code>new jobName options</code>	This action creates a new VSDB object with the specified <i>jobName</i>. For information on <i>options</i>, see “VSDB Options” on page 243. The new action is the only place you can establish the parameters of the VSDB object with the job. If you want to change parameters, such as trying different <code>-match</code> criteria, you must create a new object even if it is for the same job. Each time you create a new VSDB object, you automatically free the previous one. You may have only one VSDB object open at a time.
<code>next</code>	This action retrieves records from the persistent store step by step. The number of records retrieved depends on the value you specify for <code>-keycount num_records</code>. The default value is 10.

Action	Description
free	This action frees the VSDB object. Use this action when you want to leave the VSDB session open for a period of time but you do not want to burden the Controller with an open VSDB object. Each time you create a new VSDB object, you automatically free the previous one. Similarly, when you quit the VSDB session, the VSDB object you had open is automatically freed.
quit	This action quits the VSDB session, frees any open VSDB object, and returns to a normal K2 Spider Client session.

VSDB Options

The following table contains the supported options for a VSDB session.

Option	Description
-casesen	This option specifies case-sensitivity for <i>record</i> and <i>expression</i> values.
-delcoll	This option specifies the records to mark for deletion from the collection. You can specify this option only once in each command line. To mark more than one type of document, run multiple instances of vsdb. You may want to also use -showrec to view the document keys that are deleted.
-delete	This option specifies the records to mark for deletion from the persistent store, based on the supplied expression with one of -match, -key, or -status. You can specify this option only once in each command line. To mark more than one type of document, run multiple instances of vsdb. When you use -delete, you must also use one of the -match, -key, or -status switches. You might want to use this if you want to keep the current versions of the documents in the collection for searching without having them updated. By marking the records for deletion in the persistent store, the documents will not be updated or refreshed. If you want to mark the document key for deletion from the collection as well, use -delcoll.

Option	Description
<code>-match <i>expression</i></code>	This option counts records that match the specified expression. If you also use <code>-regexp</code>, you may specify a regular expression. If you do not use <code>-regexp</code>, then you can only use a wild-card expression. You cannot use multiple expressions, such as: <pre>-match '*html' -match '*pdf'</pre> You can use this option as criteria for the <code>-delcoll</code> option.
<code>-key <i>key</i></code>	This option reports whether the specified key exists. To have the matching key printed to standard output, use this option in conjunction with <code>-showrec</code>. You can use this option as criteria for the <code>-showrec</code> or <code>-delcoll</code> options.
<code>-keycount <i>num_records</i></code>	This option specifies how many records to display when you use the next action. If you do not use <code>-keycount</code> to specify a value, one record will be shown with each next action. The default value is 10.
<code>-regexp</code>	This option specifies the use of regular expressions rather than the default wild-card expressions.

Option	Description																
-showrec	This option specifies that records determined by the -match, -key or -status criteria are to be displayed. You may want to use this option in conjunction with -delete or -delcoll in order to see which records you are actually deleting.																
-status <i>status</i>	Counts records of document keys matching the specified status code(s). Although you are querying about document key status, what you will get back is actually a record of that status from the persistent store. Use the -showrec option to have the matching records also printed to standard output. Multiple status codes are combined with the Boolean AND operator. Choose from: <table><tr><td>none</td><td>document key is not being processed</td></tr><tr><td>cand</td><td>document key is a candidate for indexing</td></tr><tr><td>used</td><td>document key is being fetched</td></tr><tr><td>ins</td><td>document key is in inserting queue</td></tr><tr><td>upd</td><td>document key is in update queue</td></tr><tr><td>del</td><td>document key is in delete queue</td></tr><tr><td>done</td><td>document key has been processed</td></tr><tr><td>fail</td><td>document key fetching has failed</td></tr></table> The lifecycle of a document key in the persistent store is: new URL > cand > used > ins/upd/del > done. A successful indexing job will result in document keys in either a none or done status. If you notice many document keys in any of the other states, it is likely the indexing job was interrupted. You should restart the job so that document keys with a status of cand, used, ins, upd, del, or fail are processed.	none	document key is not being processed	cand	document key is a candidate for indexing	used	document key is being fetched	ins	document key is in inserting queue	upd	document key is in update queue	del	document key is in delete queue	done	document key has been processed	fail	document key fetching has failed
none	document key is not being processed																
cand	document key is a candidate for indexing																
used	document key is being fetched																
ins	document key is in inserting queue																
upd	document key is in update queue																
del	document key is in delete queue																
done	document key has been processed																
fail	document key fetching has failed																

k2spider_cli Reference

At the `k2spider_cli>` prompt, type a question mark (?) to view the list of commands.

Command	Description
<code>batch batch_file</code>	Executes the file which contains these commands in a batch. For example, you could store a series of <code>k2spider_cli</code> commands in a file called <code>cmd1</code> , and then use this command: <code>k2spider_cli> batch cmd1</code>
<code>connect c_host:port</code>	Connects to the specified controller.
<code>create inifilepath</code>	Creates a new job on the controller to which the client is connected. The value for <code>inifilepath</code> must be either an absolute path or just the root name of the job if you have launched <code>k2spider_cli</code> from the directory in which the INI file is located. NOTE: Although you can specify the root name of the job initialization file, the actual file must have an extension of <code>.ini</code> . For example, for a job file named <code>myjob1.ini</code> , you can specify <code>myjob1</code> for <code>inifilepath</code> .
<code>createcoll crwl/indx crwl_or_indx_spec collname gatewaytype styledir [authfilepath]</code>	Creates a runtime collection. NOTE: Make sure you specify the appropriate <code>host:port</code> combination for <code>crwl_or_indx_spec</code> , depending on whether you use the <code>crwl</code> (for Crawler) or <code>indx</code> (for Indexer) argument.
<code>delete jobname</code>	Deletes a job on the controller to which the client is connected.
<code>encrypt password</code>	Outputs the encrypted form of the specified plaintext <code>password</code> . For forms based authentication, passwords are stored in the <code>[FormsAuth]</code> section of the job INI file. For basic authentication, the passwords are stored in a file indicated by the <code>auth</code> option in the <code>[Job]</code> section. You may store passwords in these places in clear text or in encrypted format. Use the <code>encrypt</code> command to output the encrypted format.
<code>getinfo component</code>	Reports information from the specified component. Choose from <code>ctrl</code> for the Controller, <code>crwl</code> for the Crawler and <code>indx</code> for the Indexer.
<code>getstat jobname</code>	Reports information on the specified job.

Command	Description
<code>getjobini <i>jobname</i> <i>filepath</i></code>	Copies the job INI file for the specified job to the specified file path. NOTE: Although you can specify the root name of the job initialization file, the actual file must have an extension of <code>.ini</code> . For example, for a job file named <code>myjob1.ini</code> , you can specify <code>myjob1</code> for <i>jobname</i> .
<code>help or ?</code>	Displays this help file.
<code>modify <i>inifilename</i></code>	Submits the modified INI file to the controller. NOTE: Although you can specify the root name of the job initialization file, the actual file must have an extension of <code>.ini</code> . For example, for a job file named <code>myjob1.ini</code> , you can specify <code>myjob1</code> for <i>inifilename</i> .
<code>pause <i>jobname</i></code>	Pauses the specified job.
<code>purge <i>jobname</i></code>	Purges records from the job's persistent store and the document keys from the collection.
<code>q or quit</code>	Quits the K2 Spider Client session.
<code>resume <i>jobname</i></code>	Resumes the paused job.
<code>start <i>jobname</i></code>	Starts the specified job.
<code>stop <i>jobname</i></code>	Stops the specified job.
<code>shutdown <i>component</i></code>	Shuts down the specified component. Specify <code>ctrl</code> to shut down the controller and everything registered with it. Specify a crawler or indexer <i>host:port</i> to shut down an individual component.
<code>submit bif <i>bulk_insert_file</i> <i>jobname i_host:port</i> <i>submitsize</i> insert update modify delete</code>	Submits the bulk insert file (BIF) to the job on the specified indexer. Specify a <i>submitsize</i> value and whether you are inserting or updating.
<code>submit doc <i>filepath</i> <i>jobname [i_host:port]</i></code>	Submits the document to the specified job. Specifying an indexer <i>i_host:port</i> is optional.
<code>submit keys <i>filelist</i> <i>jobname [i_host:port]</i></code>	Submits the document keys in the filelist to the specified job. Specifying an indexer <i>i_host:port</i> is optional.
<code>vsdb</code>	Enters into a VSDB session. NOTE: This is not the same as the <code>vsdb</code> command-line tool available in the <code>/bin</code> directory for Verity Spider.

Command	Description
<code>getenv envVarName</code>	Gets the contents of the specified environment variable. Use at the direction of Verity Technical Support only.
<code>putenv envVarName envVarValue</code>	Writes the specified value to the specified environment variable. Use at the direction of Verity Technical Support only.

Job Initialization File Reference

The job initialization file contains information to determine the behavior, timing and parameters of an indexing job.

This chapter includes the following sections:

- General Conventions
- Summary of INI File Options
- Job Section
- FormsAuth Section
- Crawlers Section
- Indexers Section
- Network Section
- Schedule Section
- Proxy Section
- DocRouter Section
- Criterion Section
- Sample myjob.ini File
- Evaluating “include” and “exclude” Criteria
- The Use of Last-Modified Date

General Conventions

Following are the general conventions for the contents of the `job.ini` file.

- Comment lines start with a semi-colon (;)

For example:

```
; JOB SECTION
```

- In-line comments start with a semi-colon (;) at any location in a line

For example:

```
start = ;include domains in secure URLs
```

- Each section starts with a section name in square brackets

For example:

```
;*****  
[Job]
```

- Attribute names and values must be separated by an equal sign (=)

For example:

```
jobname = myjob
```

- Multiple attributes are specified by repeating the name = value line

For example:

```
; INDEXER SPEC SECTION  
[Indexers]  
indexerspec = caliber:9090  
indexerspec = caliber:9091
```

- Attribute names are not case-sensitive

For example, the following entries are identical.

```
jobname = myjob
```

and

```
jobname = MYjob
```

- Extra spaces and tabs are allowed and ignored

For example:

```
jobname = myjob
```

Spaces are added in the sample file for better readability.

- Valid attributes values are provided as in-line comments in this file

For example:

```
loglevel = verbose ;nostdout|skip|summary|verbose|debug|trace
```

- To preserve apostrophes ('), enclose the word that contains the apostrophe in double quotes.

For example:

```
jobname = "quinn'sjob"
```

- To preserve spaces, enclose the word or phrase that contains the spaces in double quotes.

For example:

```
rule = localhost:9999 "breaking news"
```

If you want to keep the words together, for a phrase search, for example, enclose the phrase in single quotes within double quotes as follows:

```
rule = localhost:9999 "'breaking news'"
```

Summary of INI File Options

The following tables identify the sections and the options which are specified in each. You can use these tables to quickly identify the contents of each section.

A sample job INI file is installed for you with K2 Spider in:

installDir/k2/samples/cmdfiles/k2spider/myjob.ini

where *installDir* is the directory in which you installed K2 Spider.

Table 9-1 [Job] Section Options

abspath	auth	casesen	collectanchortext
collectlinkinfo	cookiefile	gatewaytype	jobname
loglevel	maxnumdocs	maxnumdocsperhost	metamap
mimemap	nocache	nosubmit	prefixmap
processbif	regex	reparse	repos_charmap
start			

Table 9-2 [FormsAuth] Section Options

LoginURL	Action	ErrorURL	Method
Charset	any_field		

Table 9-3 [Crawlers] Section Options

crawlerspec

Table 9-4 [Indexers] Section Options

indexerspec

Table 9-5 [Network] Section Options

agentname	delay	header	retry
timeout			

Table 9-6 [Schedule] Section Options

schedmode	year	month	day
hour	minute	frequency	repeat

Table 9-7 [Proxy] Section Options

hostname	portnum	username	password
noproxy			

Table 9-8 [DocRouter] Section Options

routertype	rule	rulefile

Table 9-9 [Criterion] Section Options

cgiok	detectdupfile	domain	exclude
followdup	followsymlink	host	https
ignoreacl	include	indexclude	indinclude
indmimeexclude	indmimeinclude	indskip	javascriptok
jumps	lastmodified	maxdocsize	mimeexclude
mimeinclude	mindocsize	nodocrobo	nodupdetect
nofollow	norobo	pathlen	query
refreshtime	skip	unlimited	

Job Section

The Job section [Job] is where you specify options which affect how the job is run and how documents are treated when they are gathered.

abspath

Description Generates absolute paths for files.

Type File system only (`gatewaytype` = `filesystem`)

Syntax `abspath` = [true | false]

Details Use this option when the document locations are not going to change, but the collection might be moved around.

When you index a Web server's contents through the file system, you should use `prefixmap` with `abspath` to map the absolute file paths to URLs.

For more information, see [“Prefix Mapping” on page 352](#).

Default Absolute paths are not generated. By default, Verity Spider stores paths relative to the location of the collection into which documents are being indexed. You must set `abspath` = `true` to store absolute paths.

See Also [prefixmap](#)

auth

Description Specifies an authorization file to support authentication for secure Web indexing.

Type Web crawling only (`gatewaytype = web`)

Syntax `auth = absolute_path_and_filename`

Relative paths are not supported. You must specify the absolute path.

Details Some Web sites secure documents by requiring authentication on the directory in which they are stored. This is typically accomplished using the administration functions of the Web server. In order to retrieve documents for indexing from such secured directories, or realms, the K2 Spider uses the `auth` option to reference an authorization file. The authorization file contains one record per line. Each line consists of server, realm, user name, and password, separated by white space.

See “[Indexing Secure Web Sites](#)” on page 339 for additional information on completing the following steps.

1. Use `auth` to index a realm. A realm is a logical boundary set in a Web server to differentiate authentication credentials and authorized file or URL access. You can have multiple realms on a single Web server. The realm value appears in the browser’s user name-password prompt. Use the K2 Spider Client option `encrypt` to create an encrypted version of the password.
2. (Optional) Use the K2 Dashboard to create an `auth` file.

Note When you specify starting point URLs, you can optionally specify a user name and password for access to the specified URL. K2 Dashboard automatically creates a file called `vgwhttp.auth` that contains those credentials.

3. Add the starting points to the `job.ini` file.
4. Edit the `vgwhttp.prm` file.

Note Copy the `vgwhttp.prm` file in `common/style/httpsec` directory to another location and use it as an example.

Because the K2 Spider automatically detects if the password is encrypted, there is no need to use the `-auth2` option with K2 Spider.

Default An authorization file is not specified.

casesen

Description Specifies that processing is case-sensitive.

Syntax `casesen = [true | false]`

Details When `casesen = true`, this option makes processing case-sensitive by specifying that the K2 Spider process separately document keys that differ only in case. Use only for indexing UNIX servers.

Default Processing is not case-sensitive.

collectanchortext

Description Specifies that text between an opening `<a href="">` and closing `</a>` tag is indexed into a special zone and appended to the document referred to within the `<a href="">` element.

Type Web crawling only (`gatewaytype = web`)

Syntax `collectanchortext = [true | false]`

Details When `collectanchortext = true`, the Crawlers specified in this job INI file traverse all available links, as defined by other criteria in this job INI file, to gather the text for all `<a href="">` elements found within all indexable documents.

For more information on how this anchor text is used, see the *Verity Query Language and Topic Guide*.

Default Anchor text is not gathered.

Note `collectanchortext` and `maxnumdocs` cannot be used together.

Warning! If `collectanchortext = true`, indexing cannot start until all reachable documents for the job have been gathered and their `<a href="">` elements parsed. This is necessary because all of the links to all documents cannot be determined until all reachable documents are available. This may significantly delay when indexing occurs.

collectlinkinfo

Description Specifies that link information is gathered by the Crawlers defined in this job INI file.

Type Web crawling only (`gatewaytype` = web)

Syntax `collectlinkinfo` = [true | false]

Details When `collectlinkinfo` = true, the Crawlers traverse all available links, as defined by other criteria in this job INI file, to gather link information for all documents within the scope of the criteria. This link information is used while indexing the documents to improve search relevancy.

For more information on how this link information is used, see the *Verity Query Language and Topic Guide*.

Default Link information is not gathered.

cookiefile

Description Specifies the location of a Netscape-style cookie file.

Syntax `cookiefile` = *path_and_filename*

Details The `cookiefile` option enables K2 Spider to support multiple cookies per site by referring to an external cookie file. When `cookiefile` is specified, K2 Spider will open the file specified by *path_filename* and import the cookies it contains.

Default A cookie file is not specified.

Note Netscape-style cookies are supported, Internet Explorer-style cookies are not.

gatewaytype

Description Specifies the type of gateway employed by the Indexer(s).

Syntax gatewaytype =
[notes|odbc|documentum|exchange|web|filesystem|gateway]

The gateway value is reserved for OEM customers using the Gateway Developers Kit (GDK). For more information, see the *Verity OEM Deployment Guide*.

Details This option specifies the type of gateway employed by the Indexer(s) you specify in the INDEXER SPEC (see [“Indexers Section” on page 271](#)). The value you specify here must be the same as the value for -gateway specified for the Indexer which will be servicing this job. See [“Running an Indexer” on page 193](#).

For example, if you specify a gateway type of Web for the job in the INI file,

```
[Job]
gatewaytype = web
```

and you specify the indexer as follows,

```
; INDEXER SPEC SECTION
[Indexers]
indexerspec = CAIRO:9090
```

then the Indexer on CAIRO must be started as follows:

```
k2spider_srv -indexer CTRL:9090 -port 9091 -gateway web
```

You may specify multiple Indexers in the INDEXER SPEC SECTION, but they must all be of the same gateway type.

Note A Crawler started with -gateway web can service Indexers with either -gateway web or -gateway filesystem. Conversely, a Crawler started with -gateway filesystem can also service Indexers with either -gateway web or -gateway filesystem. For other gateway types, however, everything must be of the same gateway type. This includes Crawlers, Indexers, and the gatewaytype entry in the appropriate job INI files.

You can start a Crawler without specifying a gateway type with -gateway. When you start an Indexer with the -collection option, the gateway will be determined.

Default gatewaytype = web

See Also [“Running a Crawler” on page 184](#) and [“Running an Indexer” on page 193](#).

jobname

Description Specifies the name of the indexing job.

Syntax `jobname = jobname`

Details The job name can be the same as the root name (without the `.ini`) of the `.ini` file, although it is not required. It is required, however, that the job initialization file must have an extension of `.ini`.

Default No value.

loglevel

Description Specifies the types of messages to log.

Syntax `loglevel = [nostdout] argument`

Details If you add `nostdout` to the `loglevel` argument, messages will not be written to standard output. Log files, however, will still be created.

Note K2 Spider does not allow its log files to grow beyond 500 MB. When a log file reaches 500 MB, it is backed up as `FileName.old`. When the new file grows to 500 MB, it is renamed to `FileName.old`, overwriting the previous `FileName.old`. Thus an administrator can see only the most recent 500MB - 1 GB of log data.

Default The default log level is `skip`. Messages are written to standard output and to various log files in the subdirectory `workpath/jobname/log`, where `workpath` is specified by `-workpath` (for `k2spider_srv`) and `jobname` is the name of the job.

Choose one of the following arguments to determine which message types are logged.

Loglevel arguments	Description
<code>skip</code>	This is the default log level for <code>k2spider_srv</code> . Includes the following message types: warning, error, badkey, skip Use this option only if you do not want summary type messages.
<code>summary</code>	Includes the following message types: information, warning, error, badkey, progress, summary Use this option only if you do not want skip type messages.
<code>verbose</code>	Includes the following message types: information, warning, error, badkey, progress, summary, skip

Loglevel arguments	Description
debug	Includes the following message types: information, warning, error, badkey, progress, summary, skip, debug Note: This argument should be used only at the direction of Verity technical support or for troubleshooting indexing problems.
trace	Includes the following message types: information, warning, error, badkey, progress, summary, skip, debug, trace Note: This argument should be used only at the direction of Verity technical support or for troubleshooting indexing problems.

maxnumdocs

Description Specifies the maximum number of documents to be crawled.

Syntax `maxnumdocs = num_docs`

Details As soon as the number of crawled documents reaches the specified number, the Controller signals the Crawlers to stop. The value for `num_docs` does not necessarily correspond exactly to the number of documents indexed. Some of the documents submitted to the Indexers may fail to index because they might be invalid or corrupt.

Default There is no maximum specified.

Note `maxnumdocs` and `collectanchortext` cannot be used together.

maxnumdocsperhost

Description Specifies the maximum number of documents to be downloaded or submitted for indexing from each host encountered during indexing jobs.

Type Web crawling only (`gatewaytype = web`)

Syntax `maxnumdocsperhost = num_docs`

Details This option can be useful for running ‘discovery’ indexing jobs where you specify a low number for `num_docs` and open up other constraints (such as number of jumps). By using `domain` and other such options carefully, you can crawl your entire enterprise and hit as many hosts as possible while only indexing a few documents per host.

Default 500,000 documents.

metamap

Description Enables you to use a text file to map custom meta tags to valid HTTP header fields.

Type Web crawling only (`gatewaytype = web`)

Syntax `metamap = path_and_filename`

Details This option enables you to use a text file to map custom meta tags to valid HTTP header fields.

This means you are able to use your own meta tag in the document to replace what is returned by the Web server, or to insert it if nothing is returned. Currently, the only header fields of real value are:

- Last-Modified

The syntax for a Last-Modified entry in the text file is:

```
your_meta_tag Last-Modified y/n
```

- Content-Length

The syntax for a Content-Length entry in the text file is:

```
your_meta_tag Content-Length y/n
```

where *y/n* is an override flag which can be either yes (y) or no (n).

Default No file is specified.

Example A mapping file for `-metamap` might include:

```
Doc_Last_Touched Last-Modified n
Doc_Size Content-Length y
```

In this example, you want to replace the standard Last-Modified meta tag with your own meta tag, `Doc_Last_Touched`. Similarly, you want `Doc_Size` to replace Content-Length.

If you use the *y* override flag, the custom meta tag value overrides the value for the valid field, even if both values are present and different. This can be useful when the valid field value is always sent, but you want to specify your own value with a custom meta tag.

If you use the *n* override flag, then the value for the custom meta tag will be used only if there is no value for the valid field returned by the server. If a value for the valid field exists, then that is given precedence.

Warning! If you have several entries mapping to the same valid field, only the last entry will take effect.

mimemap

Description Specifies a control file (simple ASCII text) that maps file extensions to MIME Types.

Syntax `mimemap = path_and_filename`

Details This option enables you to make custom associations and to override defaults.

The format for the control file is:

#file_ext_no_dot	MIME Type
abc	application/word

Default No file is specified.

nocache

Description Disables the caching of files during Web site indexing.

Type Web crawling only (`gatewaytype = web`)

Syntax `nocache = [true | false]`

Details Use this option with `nosubmit` or when the Indexer is executed with the `-noindex` option. This has the effect of decreasing the demands on your disk space.

Normally, the Crawler downloads URLs and writes them to a bulk insert file while downloading the documents themselves. When indexing occurs, once `submitsize` (as specified by the `-submitsize` Indexer option) has been reached, the cached files are indexed and then deleted. If you use `-noindex`, the bulk insert file is submitted but not processed by K2 Spider, and so the documents are not deleted until indexing occurs. This will usually be with `mkvdk`. For information on `mkvdk`, see the *Verity Collection Reference*.

By using `nocache` in conjunction with `nosubmit`, you avoid storing files locally at all. Files are downloaded only when indexing actually occurs.

Default Files are cached during Web site indexing.

See Also `nosubmit`

nosubmit

Description Specifies that K2 Spider gather document locations without indexing them.

Syntax `nosubmit = [true | false]`

Details The document locations are stored in a Bulk Insert File (BIF), which is not submitted to the collection. This option is typically used in conjunction with a separate indexing service, such as `mkvdk`. Note that with an indexing service other than K2 Spider, you must specify the name and path for the BIF because the collection has no record of it.

Default `false`

See Also `nocache`

prefixmap

Description Specifies a control file (simple ASCII text) that maps one field to another.

Syntax `prefixmap = path_and_filename`

Details This option is typically used to create a URL field that is the Web equivalent of a file system path. File system indexing is faster than web crawling over the network. Using `prefixmap` to replace the file system path with the URL means relative hyperlinks in the HTML pages are kept intact when viewed through K2 Server.

A scenario is you want to index from a web server is limited to the files available by way of a file system rather than through the web server itself.

The format for the control file is:

```
src_field src_prefix dest_field dest_prefix [/]
```

where the optional slash (/) indicates that Windows file system backslashes need to be translated into URL-friendly forward slashes.

For example, to map the file path `/usr/pub/docs` to `http://web/~verity`, use:

```
vdkgwkey /usr/pub/docs URL http://web/~verity
```

For example, to map the file path `c:\pub\docs` to `http://web/~verity`, use:

```
vdkgwkey c:\pub\docs URL http://web/~verity /
```

Note the use of the slash (/) due to the file system path backslashes (\).

See Also `abspath` and [“Using the Control File” on page 352](#).

processbif

Description Specifies a path to a script that the Indexer calls to operate on the temporary BIFs that it generates.

Syntax `processbif = path_to_script`

Description When a Crawler gathers document locations, it sends those locations as document keys in a Verity Bulk Insert File (BIF) to a Indexer. The number of keys in each BIF is determined by the `-submitsize` Indexer option when the Indexer is launched.

Normally, when a Indexer not running in `-noindex` mode receives a BIF, it processes it immediately and the document to which the keys in the BIF refer are indexed.

With `processbif`, you can call a script to add information such as meta data to the BIF before it is processed for indexing by the Indexer.

Default No file is specified.

Note You can only add information to the BIFs that are automatically generated by the Indexer.

Example Using a script to add author information to the document keys in the BIFs as they are generated.

On UNIX, the script is `addauthormetadata.sh`:

```
processbif = ./path/to/addauthormetadata.sh
```

With the contents of `addauthormetadata.sh`:

```
#!/bin/sh
/usr/bin/java -classpath ./path/to/myclass
/path/to/myclass/myprocbifclass '%1'
# Arg %1 is the path to the BIF
# Enclose %1 in quotes to account for possible spaces in path
```

On Windows, the script is `addauthormetadata.bat`:

```
processbif = "%path\to\addauthormetadata.bat"
```

With the contents of `addauthormetadata.bat`:

```
perl "E:\colls\process_bif\hello.pl" "%!*%"
```

The string `!*` is a placeholder for the BIF path. As it may contain spaces, it should also be enclosed in escaped double quotes as shown.

regexp

Syntax `regexp = [true | false]`

Details When `regexp = true`, this option specifies the use of regular expressions rather than the default wild-card expressions for the following options: `exclude`, `indexinclude`, `include`, `indinclude`, `skip`, `indskip`, `preferred`, and `nofollow`.

Wild-card expressions allow the use of the asterisk (`*`) for text strings, and the question mark (`?`) for single characters.

This wild-card expression...	Will apply to these text strings...
<code>a*t</code>	although, attitude, audit
<code>file?.htm</code>	files.htm, file1.htm, filer.htm
<code>name?.*</code>	names.txt, name.doc, named.blank, names.ext

Regular expressions allow more flexible means for matching alphanumeric strings. For example, to match "ab11" or "ab34" but not "abcd" or "ab11cd," you could use the following regular expression:

```
^ab[0-9][0-9]$
```

Default Only wild-card expressions can be used.

reparse

Description Forces parsing of all HTML documents already in the collection.

Type Web crawling only (`gatewaytype = web`)

Syntax `reparse = [true | false]`

Details You must specify a starting point with the `start` option when you use `reparse`. The `reparse` option enables you to retrieve new documents that are one level or more below documents you have previously indexed. For example, you may have a file named `toc.htm` which links to `1.htm`, `2.htm`, `3.htm` and `4.htm`. If any of the numbered files change, subsequent indexing jobs will not see them because `toc.htm` will be ignored due to the fact that it was already indexed. By using `reparse`, you gain access to those linked files.

You can also use `reparse` to include paths and documents which were previously skipped due to exclusion or inclusion criteria. In this case, remember to change the criteria, otherwise there will be little for K2 Spider to do.

Default Behavior is equal to `false`.

repos_charmap

Description Specifies the character map name of the operating system so that the file path in the `VdkVgwKey` field can be converted to the character map of the collection.

Type Web crawling (`gatewaytype = web`) and file system (`gatewaytype = filesystem`) only

Syntax `repos_charmap = oscharmapname`

Details The file path and name to be indexed may contain special characters and be in a particular character map (of the operating system). Use this option to have K2 Spider convert the character map of the file path and name (as it is stored in the `VdkVgwKey` field) from the operating system character map to the character map of the collection into which the file is being indexed.

When you specify a value for `repos_charmap` and are indexing with the File System Gateway, K2 Spider writes the `DOC_FN` key into its Bulk Insert Files (BIFs) with the file path in encoded form. Normally the `DOC_FN` key is not present in the BIFs. Specify `repos_charmap = *` if you want to force K2 Spider to write the `DOC_FN` keys as encoded strings when indexing with the File System Gateway.

Example You are indexing a file system where files contain SJIS characters in their names. Specify `repos_charmap = sjis` in the job INI file.

start

Description Specifies a starting point for indexing. There is no limit on the number of starting points that can be listed, but each starting point must be on a separate line.

Syntax `start = path_or_URL`

Details To specify multiple starting points, add multiple lines as follows:

```
start = http://server1.name.com/point1
start = http://server1.name.com/poin2
```

For this repository type...	The starting point is...
Web	The URL or URLs from which the K2 Spider is to begin indexing. Use other options such as <code>jumps</code> to control how far from the starting point the K2 Spider goes.
File system	The starting directory or directories in which the K2 Spider starts indexing. All subdirectories beneath the starting point are indexed unless you use <code>pathlen</code> or any of the inclusion or exclusion criteria. NOTE: The user account must have all necessary access rights to any mapped drives or UNC paths you index. For information on indexing mapped network drives and UNC paths in Windows, see “Indexing Network, UNC and Secure Paths on Windows” on page 348.
Notes, ODBC, Documentum, Exchange	The value specified for <code>start</code> acts as a filter on the starting points specified in the gateway’s configuration file. Generally, this is most useful for a refresh where you only want to update specific starting points within the configuration file. For these repository types, and consequently those gateways, the value for <code>start</code> is a <code>VdkVgwKey</code> field value. For more information on the format of a <code>VdkVgwKey</code> value for a particular gateway, and for any particular setup information you may need, see the documentation for that gateway. A value for the <code>start</code> option is not required. If there is no value, K2 Spider will use the information in the gateway configuration file to determine what to crawl.

Default No value.

FormsAuth Section

The FormsAuth section [FormsAuth] is where you specify details of how forms-based authentication is to be done when indexing web sites that have forms for authentication. You can specify multiple [FormsAuth] sections in a job INI file for multiple forms. Each [FormsAuth] section for a given form must have corresponding entries in the HTTP Gateway configuration file, `vgwhttp.cfg` file, to enable viewing of protected pages. See the *Verity Collection Reference* for information on what is required in `vgwhttp.cfg`.

LoginURL

Description Identifies the page that collects credentials.

Syntax `LoginURL = URL`

Example `LoginURL=http://www.mycompany.com/finance/login.asp`

Action

Description Specifies the page that authenticates the user with the credentials supplied in the Username and Password entries.

Syntax `Action = URL`

Example `Action=http://www.mycompany.com/finance/submit.asp`

ErrorURL

Description Specifies the page that is used to display authentication errors to the user.

Syntax `ErrorURL = URL_for_login_errors_page`

Example `ErrorURL=http://www.mycompany.com/finance/loginError.asp`

Method

Description Specifies whether the GET or POST method is used with the Action URL for the purpose of authentication.

Syntax Method = [GET | POST]

Charset

Description Specifies the character set of the username and password.

Syntax Charset = *character_set*

any_field

Description Specifies the name of a field expected by the form. Specify as many as are required.

Syntax *any_field* = *value*[*;encrypted*]

Details For example, if you need a user name and password for authentication, specify *username=me* and *password=mypass*. You may add an optional *;encrypted* comment at the end of a line to indicate that the value of that attribute is encrypted. Use the K2 Spider Client [encrypt](#) command to encrypt values.

See Also [“encrypt” on page 214](#) and the *Verity Collection Reference* for information on what is required in the HTTP gateway configuration file, *vgwhttp.cfg*.

Crawlers Section

The Crawler section [Crawlers] is where you specify the Crawlers which will be responsible for gathering document locations for the indexing job.

crawlerspec

Description Specifies the Crawlers for the job.

Syntax `crawlerspec = host:port`

Details You can specify multiple Crawlers by repeating the `crawlerspec` entry for each Crawler. A maximum of ten Crawlers and Indexers can be registered to each Controller.

Following is an example [Crawlers] section.

```
;*****  
; Crawler Spec Section  
[Crawlers]  
crawlerspec = cairo2:9908
```

Crawlers must be started with the `-gateway` option. A Crawler started with `-gateway web` can service Indexers with either `-gateway web` or `-gateway filesys`. Conversely, a Crawler started with `-gateway filesys` can also service Indexers with either `-gateway web` or `-gateway filesys`. For Notes, Exchange, ODBC, or Documentum, however, everything must be of that particular gateway type. This includes Crawlers, Indexers, and the `gatewaytype` entry in the appropriate job INI files.

For more information, see [“Running a Crawler” on page 184](#).

Default There are no values in a default job INI file. If you do not specify any Crawlers, the Controller to which you send the job (associated with the current job INI file) will dispatch the job to whatever Crawlers are bound to it (the Controller). Controllers and Crawlers can be bound when you execute them. For example, you have to specify a Controller when you start the Crawler.

Indexers Section

The Indexer section [Indexers] is where you specify the Indexers which will be responsible for gathering document locations for the indexing job.

indexerspec

Description Specifies the Indexers for the job.

Syntax `indexerspec = host:port`

Details You can specify multiple Indexers by repeating the `indexerspec` entry for each Indexer. A maximum of ten Crawlers and Indexers can be registered to each Controller.

Following is an example [Indexers] section.

```
;*****  
; Indexer Spec Section  
[Indexers]  
indexerspec = cairo2:9909
```

Indexers must be started with the `-gateway` option. Whatever value is specified for `-gateway` must be matched by the `gatewaytype` entry in the job INI file. For more information, see [“Running an Indexer” on page 193](#).

Default There are no values in a default job INI file. If you do not specify any Indexers, the Controller to which you send the job (associated with the current job INI file) will dispatch the job to whatever Indexers are bound to it (the Controller). Controllers and Indexers can be bound when you execute them. For example, you have to specify a Controller when you start the Indexer.

Network Section

The Networking section [Network] is where you specify options specific to traversing a network.

agentname

Description Specifies the value for the agent name field that is part of the HTTP request.

Type Web crawling only (`gatewaytype = web`)

Syntax `agentname = string`

Details Since Web servers can be configured to return different versions of the same page depending on the requesting agent, you can use `agentname` to impersonate a browser client. Use double-quotes if the name contains a space.

Default Internally, this is set to `vspider`.

delay

Description Specifies the minimum time between HTTP requests in milliseconds.

Type Web crawling only (`gatewaytype = web`)

Syntax `delay = num_milliseconds`

Default 0 milliseconds for no delay.

header

Description Specifies an HTTP header to be added to the spidering request.

Type Web crawling only (`gatewaytype = web`)

Syntax `header = string`

Details For example:

```
header = "Referer: http://www.verity.com/"
```

K2 Spider sends some predefined headers, such as Accept and User-Agent, by default. Special headers are sometimes necessary to correctly index a site.

Default No value.

Note Misuse of this option will cause indexing failure. In the event that this happens, re-run the indexing job with modified `header` values.

retry

Description Specifies the number of times **K2 Spider** should attempt to access a URL.

Type Web crawling only (`gatewaytype = web`)

Syntax `retry = num_retries`

Details You should use `retry` when it is likely that an unstable network connection will give false rejections.

Default 3 retries after the initial try.

timeout

Description Specifies time period, in seconds, that the **K2 Spider** should wait before timing out on a network connection and on accessing data.

Type Web crawling only (`gatewaytype = web`)

Syntax `timeout = num_seconds`

Details The data access value is automatically twice the value you specify for the network connection timeout.

Default The network connection timeout is 30 seconds, with the value for the data access timeout being 60 seconds.

Schedule Section

The Scheduling section [`Schedule`] is where you specify when to run the indexing job, and in the case of a scheduled job, how often.

schedmode

Description Specifies whether the indexing job is to run immediately or at some future time.

Syntax `schedmode = schedule`

Details When no value is specified for `schedmode`, the indexing job automatically runs immediately, and all other entries in the [`Schedule`] section are ignored. To run the job again, use the `start` command. For more information, see [Chapter 8, “Using K2 Spider Client.”](#)

When `schedmode = schedule`, the indexing job will start according to the values specified for the other options in the Scheduling section.

Default No value.

Note When you have a scheduled job, and the `k2spider_srv` components (Controller, Crawler and Indexer) for that job are stopped long enough for the next scheduled run time to pass, you will see errors the next time you start the Controller regarding the state of the job as Inactive.

For inactive jobs, you can just restart it with the `start` command to continue with the next scheduled run of the job, or you can use any of the other K2 Spider Client job management commands such as `purge` or `delete`. For more information, see [Chapter 8, “Using K2 Spider Client.”](#)

year

Description Specifies the year for the scheduled time of the indexing job.

Syntax `year = year`

Details Valid values are 2000-2037.

Default No value.

month

Description Specifies the month for the scheduled time of the indexing job.

Syntax `month = month`

Details Valid values are 1-12.

Default No value.

day

Description Specifies the day for the scheduled time of the indexing job.

Syntax `day = day`

Details Valid values are 1-31.

Default No value.

hour

Description Specifies the hour, in 24-hour format, for the scheduled time of the indexing job.

Syntax `hour = hour`

Details Valid values are 0-23.

Default No value.

minute

Description Specifies the minute for the scheduled time of the indexing job.

Syntax `minute = minutes`

Details Valid values are 0-59.

Default No value.

frequency

Description Specifies how often the indexing job is to run.

Syntax `frequency = [daily|weekly|monthly|weekday]`

Details Remember that, depending on how long an indexing job takes to complete, you may not get the exact frequency you expect.

For example, you might schedule a job to start at 6:00AM on a Monday and set the frequency to daily. You would therefore expect the job to next run at 6:00AM on Tuesday. If the job takes 25 hours to complete, the next run time for this job would be at 6:00AM on Wednesday. The expected run time for Tuesday must be skipped because the job completed after the scheduled time.

Default No value.

Note `frequency` and `repeat` are mutually exclusive. When neither `frequency` nor `repeat` are specified, the indexing job is executed only once according to the values specified for the other options in the Scheduling section.

repeat

Description Specifies the number of minutes the Indexer will wait after the indexing job has finished before next running that same indexing job.

Syntax `repeat = num_minutes`

Details This option provides a greater level of granularity than `frequency`. For example, if you have an indexing job that completes quickly and must run as often as possible, using `frequency = daily` may involve too much down time. Using something like `repeat = 10` instead would allow the job to run again much more quickly.

Default No value.

Note `repeat` and `frequency` are mutually exclusive. When neither `frequency` nor `repeat` are specified, the indexing job is executed only once according to the values specified for the other options in the Scheduling section.

Proxy Section

The Proxy section [Proxy] is where you specify information about proxy servers required for access to Web sites. You do not need to specify any values for these entries if you are not indexing Web sites; in other words, if you did not specify `gatewaytype = web`.

Note See the section [“Indexing with Proxy Servers” on page 344](#) for information on how to configure an HTTP gateway configuration file (`vgwhhttp.cfg`) so that documents indexed through a proxy server can then be viewed through that proxy server.

hostname

Description Defines the hostname for the proxy server.

Syntax `hostname = proxyhostname`

Details Include the complete host name, including the domain. For example, `winecup.verity.com` works, while just `winecup` does not.

Default No value.

portnum

Description Defines the port number for the proxy server.

Syntax `portnum = port`

Default No value.

username

Description Defines a user name for proxy server connections requiring authentication.

Syntax `username = name`

Default No value.

password

Description Defines the password for proxy servers requiring authentication.

Syntax `password = password`

Default No value.

See Also [noproxy](#) for hosts which you know are accessible without having to go through a proxy server.

noproxy

Description The `noproxy` option specifies that the K2 Spider directly access the hosts whose names match those specified.

Syntax `noproxy = name_n`

Details By default, when you specify a proxy server with `hostname`, the K2 Spider first tries to access every host with the proxy information. To improve performance, use `noproxy` for those hosts you know can be accessed without a proxy host. For the *name* variable, you can use the asterisk (`*`) wild-card for text strings. For example:

```
'*.verity.com'
```

You cannot use the question mark (`?`) wild-card, nor can you use regular expressions with this option even if you have specified the `regexp` option.

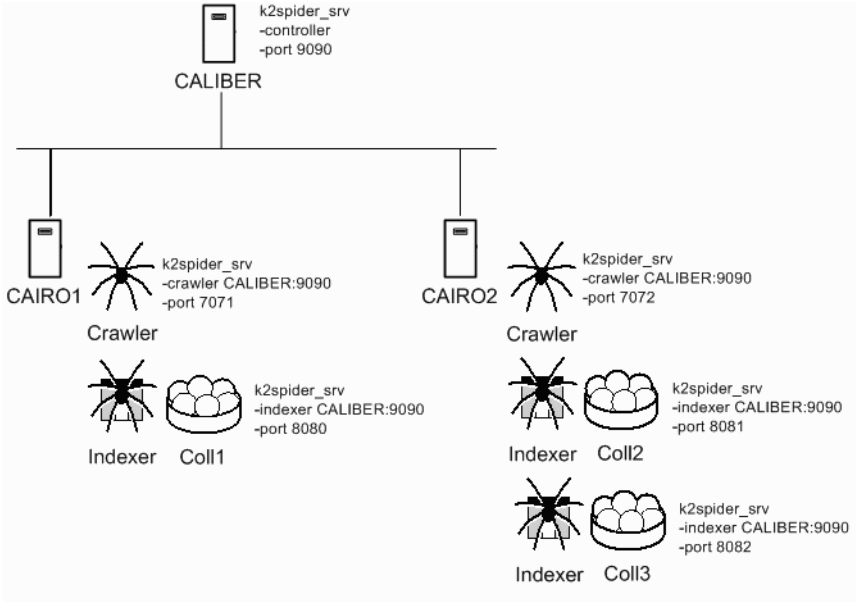
Default No value.

DocRouter Section

The Document Routing section [DocRouter] is where you define the criteria for a particular Indexer to process documents from a particular Crawler.

For example, a K2 Spider system might look like the following.

Figure 9-1 Distributed K2 Spider system candidate for document routing



routertype

Description Specifies one of the following router types: [RoundRobin](#) (Default value), [BoundCrawlIndx](#), [KeyBased](#), [MimeBased](#) or [ProfileBased](#).

Syntax `routertype = rtype`

Details The `routertype` determines how documents are fed to the Indexers.

Default `RoundRobin`.

RoundRobin

In `RoundRobin` routing, new documents discovered during crawling are distributed to each of the listed collections (through the Indexers associated with the collections), alternating among the collections. Updated documents are indexed into the same collection to which they were originally indexed.

From Figure 9-1, when a document is passed to the Crawler at port 7072 on CAIRO2, it is given to the Indexer at port 8081 on CAIRO2. As subsequent documents are passed to CAIRO2, they are distributed in a round-robin fashion between the two Indexers. Documents are never passed to the Indexer on CAIRO.

BoundCrawlIndx

In `BoundCrawlIndx` routing, all Crawlers and Indexers on the same machine are bound to each other, with round-robin distribution among the Indexers on that machine. This type of routing can reduce network traffic because only the Crawlers and Indexers on the same machine communicate.

Assume the current job being administered by the Controller CALIBER has the following [DocRouter] section in the job INI file:

```
[DocRouter]
routertype = BoundCrawlIndx
```

From Figure 9-1, when a document is passed to the Crawler at port 7071 on CAIRO, only the Indexer at port 8080 on CAIRO would index it.

Remember that all Crawlers and Indexers on the same machine are bound when `routertype = BoundCrawlIndx`. Crawlers never pass documents to a Indexer residing on another machine, but they do evenly distribute the workload to all available Indexers on the same machine.

KeyBased

In KeyBased routing, all documents matching the rule(s) specified by either the rule entry or the rulefile entry are routed to the specified Indexer. For KeyBased routing, the Controller has the following [DocRouter] section in the job INI file:

```
[DocRouter]
routertype = KeyBased
rule = indexername:port rulecriteria
```

Note You may optionally specify a rulefile entry.

For more information on rules for KeyBased routing, see "rule."

During KeyBased routing, documents are routed to all Indexers associated with each rule that matches the document keys (VdkVgwKey) as defined in the *rulecriteria*..

Using KeyBased Routing to Mirror Collections

You can use KeyBased routing to mirror collections, typically on two different hosts. Following is a sample [DocRouter] section that duplicates indexing across two Indexers with different port numbers.

```
[DocRouter]
routertype = KeyBased
rule = indexer1:9000
rule = indexer2:9001
```

This has the effect of mirroring the collections for indexer1 with those of indexer2, assuming the collections are created the same way, with the same gateway type and style set.

MimeBased

In MimeBased routing, all documents matching the MIME type(s) specified by either the rule entry or the rulefile entry are routed to the specified Indexer. For MimeBased routing, the Controller has the following [DocRouter] section in the job INI file:

```
[DocRouter]
routertype = MimeBased
rule = indexername:port rulecriteria
```

Note You may optionally specify a rulefile entry.

During MimeBased routing, matching documents are routed to only one Indexer at a time.

The following is an example of a portion of the job INI file using the information in Figure 9-1. The first rule (CAIRO2:8081) would index only HTML documents. The second rule (CAIRO2:8082) would accept and index only documents that belong to the application generic type (application/pdf, application/msword, etc.). Both rules work with both Web and file system indexing.

```
[Indexers]
indexerspec = CAIRO2:8081
indexerspec = CAIRO2:8082

[DocRouter]
routertype = MimeBased
rule = CAIRO2:8081 text/html
rule = CAIRO2:8082 application/*
```

For more information on rules for MimeBased routing, see "rule."

ProfileBased

In `ProfileBased` routing, all documents matching a given rule's criteria are routed to the specified Indexer. Documents that do not match any of the rules will be randomly routed to Indexers that are not specified in any rule (including those in a rulefile). If all of the available Indexers are specified in rules, then non-matching documents are dropped and not indexed.

For `ProfileBased` routing, the Controller has the following [DocRouter] section in the job INI file:

```
[DocRouter]
routertype = ProfileBased
rule = indexername:port rulecriteria [threshold N]
[forcethresholdvalue]
```

Note You may optionally specify a rulefile entry.

During `ProfileBased` routing, documents are routed to Indexers according to the best matched queries. Documents may be routed to multiple Indexers if `forcethreshold` is specified and the query result exceeds the threshold value (*N*).

You can specify multiple rules by repeating this entry. Here is an example rule using information from Figure 9-1:

```
rule = cairo:8081 "verity" 70
```

For more information on rules for `ProfileBased` routing, see "rule."

rule

Description Specifies the rule or rules for one of the following router types: [KeyBased](#), [MimeBased](#) or [ProfileBased](#).

Syntax `rule = indexername:port rulecriteria`

Details The rule determines how documents are fed to the Indexers.

The rule consists of two parts: an Indexer specification (*indexername:port*) and rule criteria (*rulecriteria*). You can specify multiple rule entries in the INI file, or you can use the [rulefile](#) option to refer to a text file that contains multiple rules.

- An Indexer specification

The Indexer specification consists of the name of the host on which the Indexer is running and its port number. For example, from Figure 9-1:

```
cairo:8081
```

- Rule criteria

The contents of *rulecriteria* depend on the router type.

KeyBased *rulecriteria*

For a router type of KeyBased, *rulecriteria* consists of an expression that represents a document's VdkVgwKey field value. For HTTP indexing, this will be a URL, and for file system indexing, this will be a file path.

The matching expression can be either a regular expression or a wild-card (* or ?). Documents with names matching the expression are routed to the specified indexer.

MimeBased *rulecriteria*

For a router type of MimeBased, *rulecriteria* consists of an expression.

The matching expression can be either a MIME Type name or a generic MIME Type expression in the form of *generic_type/**. For example: "text/*".

When indexing ZIP/PST documents, you must specify rules for ZIP/PST types or they will be skipped. If you want to index a ZIP file that contains Word and PDF documents, but only include MimeBased rules for those types and not the ZIP type, you will not index the Word and PDF documents in the ZIP file.

ProfileBased *rulecriteria*

For a router type of ProfileBased, *rulecriteria* consists of either a query and an optional threshold plus an optional force threshold value, or a language query.

- A query, an optional threshold value and an optional force threshold value.

The query is a Verity Query Language (VQL) query expression.

Note When your query consists of more than one word, use double-quotes (" "). For example, "verity information" is a two word query.

The threshold value is the minimum score for a document for it to be considered for the routing rule. A valid value is any number between 1 and 100. The higher the value, the more precisely the documents will match the specified query.

The default threshold value is 4.

The force threshold value is a minimum score between 1 and 100 for a document for it to be considered for the routing rule when otherwise the document would not be considered because another rule was matched with a higher score. The force threshold value is considered when there is more than one rule.

Given several rules, a document is routed to the Indexer in the rule for which the document scored highest.

For example, given the following rules:

```
rule = cairo:8081 "verity" 70
rule = cairo:9090 "other" 80
```

A document that matches the first rule with a score of 65 and the second rule with a score of 85 will be routed to the Indexer on port 9090 because the score for that rule was higher. If instead there is a force threshold value of 60 in the first rule:

```
rule = cairo:8081 "verity" 70 60
```

the document would be routed to both Indexers (port 8081 and 9090).

■ A language query

For a language-based rule, the *rulecriteria* consists of a language query as follows:

```
"VLANG <operator> langid"
```

For example:

```
"VLANG <contains> ar"
```

For *operator*, use one of *matches*, *contains* or *substring*. For more information on these operators, see the *Verity Query Language and Topic Guide*.

For *langid*, use a 2 digit language ID. For more information, see the *Verity Locale Configuration Guide*.

For documents with attachments, only the language of the parent document is considered. Regardless of the languages of the attachments, documents are routed based on the language of the parent document.

In order to use a language query for language-based routing, you must also enable the `flt_lang` filter in the `style.uni` style file. For more information on `style.uni`, see the “Syntax of style.uni File Keywords” section in the chapter titled “Document Filters and Formatting” of the *Verity Collection Reference Guide*.

rulefile

Description Specifies the location of the text file that contains the rule or rules for one of the following router types: `KeyBased`, `MimeBased` or `ProfileBased`.

Syntax `rulefile = path_to_rulefile`

Details The text file must be a simple ASCII text file that contains only rules as described for the `rule` option. All of the rules in a rule file must correspond to the same router type.

For information on rules, see the previous section, “`rule`.”

Criterion Section

The Crawling and Indexing Criteria section [Criterion] is where you specify options to control what is indexed.

cgiok

Description Enables K2 Spider to crawl URLs containing the ? symbol.

Type Web crawling only (`gatewaytype = web`)

Syntax `cgiok = [true | false]`

Details When a URL contain the ? symbol, it typically means the URL leads to a CGI or other such processing program.

The return document produced by the Web server is indexed and parsed for document links which are followed and in turn indexed and parsed. However, if the Web server does not return a page, perhaps because the URL is missing parameters required for processing to produce a page, then nothing happens. There is no page to index and parse.

Default K2 Spider will not crawl URLs containing the ? symbol, equivalent to `cgiok = false`.

Example A URL without parameters is:

`http://server.com/cgi-bin/program?`

If you include parameters in the URL to be indexed, as specified with the `start` option, then those parameters are processed and any resulting pages are indexed and parsed.

detectdupfile

Description Enables K2 Spider to detect duplicates based on a checksum when indexing file systems using the File System gateway.

Type File system only (`gatewaytype = filesystem`)

Syntax `detectdupfile = [true | false]`

Details By default, duplicate detection is disabled when indexing documents using the File System gateway. By using `detectdupfile`, a checksum is computed based on the CRC-32 algorithm. The checksum, combined with the document size, is used to determine if the document is a duplicate.

Default `false`. K2 Spider does not detect duplicate files when indexing file systems using the File System gateway.

domain

Description Limits indexing to the specified domain.

Syntax `domain = name`

Type Web crawling only (`gatewaytype = web`)

Details You must use only a complete text string for the domain name. You may not use wild-card expressions. URLs not in the specified domain will not be downloaded or parsed. You can specify multiple `domain = name` entries.

Default K2 Spider is limited to the domain specified in the values for `start`.

exclude

Description Files, paths and URLs matching the specified expression will not be followed.

Type All gateways except for ODBC Gateway (`gatewaytype = odbc`)

Syntax `exclude = expression`

Details You can specify multiple `exclude = expression` entries.

You can use wild-card expressions, where the asterisk (`*`) represents text strings and the question mark (`?`) represents single characters. For example:

```
exclude = '/my_doc*/year199?'
```

To be able to also use regular expressions with this option, you must also specify the `regexp` option.

To specify a file, path or URL which you want followed but not indexed, use `indexexclude`. For document types, use `mimeexclude` instead. For example, specify `mimeexclude application/pdf` rather than `exclude *.pdf`.

Default Nothing is explicitly excluded.

Note When specifying a URL, you must use full paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `exclude`.

followdup

Description Specifies that K2 Spider follows links within duplicate documents, although only the first instance of any duplicate document will be indexed.

Type Web crawling only (`gatewaytype = web`)

Syntax `followdup = [true | false]`

Details You may find this option useful if you use the same home page on multiple sites. By default, only the first instance of the document is indexed, while subsequent instances are skipped. If you have different secondary documents on the different sites, using `followdup` will allow you to get to them for indexing, while still indexing the common home page only once.

Default K2 Spider does not follow links within duplicate documents.

followsymlink

Description Specifies that K2 Spider follows symbolic links when indexing UNIX file systems.

Type UNIX file system only (`gatewaytype = filesystem`)

Syntax `followsymlink = [true | false]`

Default K2 Spider does not follow symbolic links.

host

Description Limits indexing to the specified host, which consists of the machine name, domain, and port number.

Type Web crawling only (`gatewaytype = web`)

Syntax `host = machinename.domain.nnn:portnumber`

Details You must use only a complete text string for the host name. You may not use wild-card expressions. URLs not on the specified host will not be downloaded or parsed. Whenever you use a port other than the default of 80, you must specify the port number. You can specify multiple `host = name` entries.

Default K2 Spider is limited to the host specified in the values for `start`.

https

Description Enables K2 Spider to index SSL-enabled Web sites.

Type Web crawling only (`gatewaytype = web`)

Syntax `https = [true | false]`

Default K2 Spider cannot index SSL-enabled Web sites.

Note You must have the Verity SSL Option installed to use `https`. The Verity SSL Option is a K2 Spider add-on available from a Verity salesperson.

ignoreacl

Description Disables the recognizing of changes to the access times for documents.

Syntax `ignoreacl = [true | false]`

Details Specify `ignoreacl = false` if you want to index documents that have changes to their access time. Keep in mind that the access time for a document can change for a variety of reasons, not just due to a change in the Access Control List (ACL) for a document.

Default K2 Spider ignores changes in document access times (`ignoreacl = true`).

include

Description Only those files, paths and URLs which match the specified expression will be followed.

Type All gateways except for ODBC Gateway (`gatewaytype = odbc`)

Syntax `include = expression`

Details You can specify multiple `include = expression` entries.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters. For example:

```
'/my_doc*/year199?'
```

To be able to also use regular expressions with this option, you must also specify the `regex` option.

If your starting points lack the specified `include` expressions, nothing will be indexed. The `include` option prevents K2 Spider from following anything which does not match the specified expressions. You may want to use the `indinclude` option instead. Where `include` prevents K2 Spider from even following anything which does not match the specified expressions, `indinclude` enables K2 Spider to follow what matches the specified expressions, while not indexing.

For document types, use `mimeinclude` instead. For example, specify `mimeinclude = text/html` rather than `include = *.htm`.

Default Nothing is explicitly included.

Note When specifying a URL, you must specify the same format as appears in the HTML hyperlink. If the link is relative, you must change it to a fully qualified URL before you can use it with the `include` option.

If you have a link that consists of `<a href=" ../path/filename.ext">`, find out what the complete URL is and specify that value. For example:

```
include="http://host.domain.com/fullpath/path/filename.ext"
```

indexexclude

Description Specifies that the files and paths that match the *expression* are not indexed, although all documents encountered are still followed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `indexexclude = expression`

Details Files and paths in URLs which match the expression are still followed. You can specify multiple `indexexclude = expression` entries.

You can use wild-card expressions, where the asterisk (`*`) represents text strings and the question mark (`?`) represents single characters. For example:

```
indexexclude = '/my_doc*/year199?'
```

To be able to also use regular expressions with this option, you must also specify the `regexp` option.

You would use this option to gather and parse certain documents, such as HTML tables of contents, to gain access to other documents for indexing while not indexing those tables of contents themselves.

Where the `exclude` option prevents K2 Spider from following anything matching the specified expressions, `indexexclude` enables K2 Spider to follow anything while only skipping that which matches the specified expressions.

For document types, use the `indmimeexclude` option instead.

Default Nothing is explicitly excluded from being indexed.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute before you can use it with `indexexclude`.

indinclude

Description Specifies that only those files and paths that match the *expression* are indexed, but all documents encountered are still followed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `indinclude = expression`

Details You can specify multiple `indinclude = expression` entries.

You can use wild-card expressions, where the asterisk (`*`) represents text strings and the question mark (`?`) represents single characters. For example:

```
indinclude = '/my_doc*/year199?'
```

To be able to also use regular expressions with this option, you must also specify the `regex` option.

Where the `include` option prevents K2 Spider from following anything not matching the specified expressions, `indinclude` enables K2 Spider to follow anything while only indexing that which matches the specified expressions.

Default Nothing is explicitly included for indexing.

Example If you want to index all documents that include “search” in the URL at `http://web.verity.com`, you cannot use:

```
[Job]
start = http://web.verity.com
[Criterion]
include = '*search*'
```

This is because the starting point does not match the `include` criteria. Instead, use `indinclude` to follow all documents (unless, of course, you have specified any of the `exclude` options) and index only those documents that match your criteria. Simply replace `include` with `indinclude` in the above example.

Note When specifying a URL, you must use full, absolute paths using the same format as appears in the HTML hyperlink. If the link is relative, you must change it to absolute to use it with `indinclude`.

indmimeexclude

Description Specifies that the MIME Types that match the *mimetype* expression are not indexed, although all documents encountered are still followed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `indmimeexclude = mimetype`

Details You can specify multiple `indmimeexclude = mimetype` entries.

Use this option to gather some documents, such as HTML tables of contents, to gain access to other documents for indexing. The `mimeexclude` option, on the other hand, prevents specified documents from being followed at all. For the *mimetype* variable, you can include the asterisk (`*`) wild-card for text strings. For example:

```
indmimeexclude = "text/*" {
```

You cannot use the question mark (`?`) wild-card, nor can you use regular expressions with this option even if you have specified the `regexp` option.

Default No MIME Types are explicitly excluded from being indexed.

See Also [“Setting MIME Types” on page 356.](#)

indmimeinclude

Description Specifies that only those MIME Types that match the *mimetype* expressions are indexed, but all documents encountered are still followed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `indmimeinclude = mimetype`

Details You can specify multiple `indmimeinclude = mimetype` entries.

The `mimeinclude` option does not allow you to index desired documents if the starting URL is not followed. For the *mime* variable, you can include the asterisk (`*`) wild-card for text strings. For example:

```
indmimeinclude = "text/*"
```

You cannot use the question mark (`?`) wild-card, nor can you use regular expressions with this option even if you have specified the `regex` option.

Default No MIME Types are explicitly included for indexing.

Example If you want to index all Word documents at `http://web.verity.com`, you cannot use:

```
[Job]
start = http://web.verity.com

[Criterion]
mimeinclude = 'application/msword'
```

This is because the starting point does not match the `mimeinclude` criteria. Now, you can use the `indmimeinclude` option to follow all documents (unless, of course, you have specified any of the exclude options) and index only those documents that match your criteria. Simply replace `mimeinclude` with `indmimeinclude` in the example above.

See Also [“Setting MIME Types” on page 356.](#)

indskip

Description Specifies that the K2 Spider is to follow and parse links but not index any HTML document that contains the text of *expression* between the starting and ending *HTML_tag*.

Type Web crawling only (*gatewaytype* = web)

Syntax `indskip = HTML_tag "expression"`

Details For multiple *HTML_tag* and *expression* combinations, use multiple instances of `indskip`.

You can use wild-card expressions, where the asterisk (*) represents text strings and the question mark (?) represents single characters. For example:

```
indskip = a "/my_doc*/year199?"
```

To be able to also use regular expressions with this option, you must also specify the `regexp` option.

Default No HTML documents are skipped from indexing.

Example To skip all HTML documents which contain the word "personnel" in the Title element, while still parsing those documents for links to other documents, use the following:

```
indskip = title "personnel"
```

For any document that contains the word `personnel` between `<title>` and `</title>`, the content of the document itself will not be indexed, but links on the page will be parsed.

Example To avoid indexing directory listing pages, while still parsing the document and path links except for link up to the parent directory, use one of the following (depending on the Web server being indexed):

For Netscape Web servers, use the following:

```
indskip = title "*Index of*"
nofollow = "*parent directory*"
```

For Microsoft Internet Information Server, use the following:

```
indskip = a "*to parent directory*"
nofollow = "*parent directory*"
```


javascriptok

Description Enables K2 Spider to crawl some JavaScript links.

Type Web crawling only (`gatewaytype = web`)

Syntax `javascriptok = [true | false]`

Details When `javascriptok = true`, K2 Spider looks for the patterns `location.href`, `window.open`, and `document.write` in the following places:

- Between `<script>` and `</script>` tags.
- Between `<head>` and `</head>` where `<meta http-equiv="content-script-type" content="text/javascript">` exists.

When any of those patterns is found, K2 Spider retrieves URLs specified therein. K2 Spider identifies JavaScript as follows:

- `location.href` — K2 Spider looks for the assigned string URL. The URL can be a web link or a file. For example:

```
window.location.href = "http://www.verity.com"
```

- `window.open` — K2 Spider looks for a URL in the first argument in the function call. The URL can be a web link or a file. For example:

```
window.open("http://www.verity.com", "navwin", "menubar=yes,  
width=600, height=300")
```

- `document.write` — K2 Spider looks for the existence of `"http://"` or `"https://"` in an argument to the function call. The URL can only be a web link and cannot be a file. For example:

```
document.write("http://www.verity.com")
```

Default K2 Spider will not crawl JavaScript links, equivalent to `javascriptok = false`.

jumps

Description Specifies the maximum number of levels from the starting URL that an indexing job can go.

Type Web crawling only (`gatewaytype = web`)

Syntax `jumps = num_jumps`

Details If you see extremely large numbers of documents in a collection where you do not expect them, consider experimenting with this option, in conjunction with the Content options, to reduce the reach of your indexing job.

Specify a number between 0 and 255. A value of 255 is equivalent to unlimited.

Default 255. This value is equivalent to there being no limit on the number of jumps.

lastmodified

Description Specifies the documents to be indexed based on when the documents were last modified.

Type All gateways except for ODBC Gateway (`gatewaytype = odbc`)

Syntax `lastmodified=timeunits`

Details The syntax for *timeunits* is: *n* day *n* hour *n* min *n* sec, where *n* is a positive integer. Spaces separate the units of time. Valid values for each unit are:

- day — valid values are 1–31
- hour — valid values are 0–23
- min — valid values are 0–59
- sec — valid values are 0–59

Only documents that were created or modified after the specified date are indexed.

Default There is no limit.

Example If you specify `lastmodified = 21 day 6 hour 5 min 30 sec`, then only those documents which were created or modified during the past 21 days, 6 hours, 5 minutes, and 30 seconds will be indexed.

Note This option takes effect after the first time an indexing job is run.

maxdocsize

Description Specifies the maximum size, in kilobytes, for documents to be indexed.

Syntax `maxdocsize = integer`

Details Any documents larger than the value specified by `maxdocsize` will be ignored.

Default 20MB.

Note This option does not apply to ZIP or PST documents.

mimeexclude

Description Specifies a MIME Type which is neither followed nor indexed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `mimeexclude = mimetype`

Details You can specify multiple `mimeexclude = mimetype` entries.

The default is to include all MIME Types. For the *mimetype* option, you can include the asterisk (`*`) wild-card for text strings. For example:

```
mimeexclude = "text/*"
```

You cannot use the question mark (`?`) wild-card, nor can you use regular expressions with this option even if you have specified the `regexp` option.

Use the `indmimeexclude` option to allow the K2 Spider to follow documents, without indexing them, to gain access to other desirable document types.

Default No MIME Types are explicitly excluded.

See Also [“Setting MIME Types” on page 356.](#)

mimeinclude

Description Specifies a MIME Type to be followed and indexed.

Type Web crawling and file system indexing only. This option can only be used with the HTTP and File System gateways.

Syntax `mimeinclude = mimetype`

Details You can specify multiple `mimeinclude = mimetype` entries.

The default is to include all MIME types. For the *mimetype* option, you can include the asterisk (`*`) wild-card for text strings. For example:

```
mimeinclude = "text/*"
```

You cannot use the question mark (`?`) wild-card, nor can you use regular expressions with this option even if you have specified the `regexp` option.

Default No MIME Types are explicitly included.

See Also [“Setting MIME Types” on page 356.](#)

mindocsize

Description Specifies the minimum size, in kilobytes, for documents to be indexed.

Syntax `mindocsize = integer`

Details Any documents smaller than the value specified by `mindocsize` are ignored.

Default There is no limit on document size.

nodocrobo

Description Specifies ROBOT META tag directives are to be ignored.

Type Web crawling only ([gatewaytype](#) = web)

Syntax nodocrobo = [true | false]

Details In HTML 3.0 and earlier, robot directives could only be given as the file `robots.txt` under the root directory of a Web site. In HTML 4.0, every document can have robot directives embedded in the META field. Use this option to ignore them. This option should, of course, be used with discretion.

Default ROBOT META tag directives are honored.

See Also [norobo](#) and <http://www.w3c.org/TR/REC-html40/html40.txt>.

nodupdetect

Description Prevents K2 Spider from detecting duplicates based on a checksum when indexing Web sites using the HTTP gateway.

Type Web crawling only ([gatewaytype](#) = web)

Syntax nodupdetect = [true | false]

Details URL-based duplicate detection is still performed. By default, duplicate detection is enabled when indexing documents using the HTTP gateway. A document checksum is computed based on the CRC-32 algorithm. The checksum combined with the document size is used to determine if the document is a duplicate. By using the `nodupdetect` option, you can disable duplicate detection when using the HTTP gateway.

Default K2 Spider detects duplicates based on a checksum when indexing Web sites using the HTTP gateway.

See Also [followdup](#)

nofollow

Description Specifies that the K2 Spider cannot follow any links specified by the `<a href="link">` tag when the *expression* occurs between `<a href="link">` and `</a>`.

Type Web crawling only (`gatewaytype = web`)

Syntax `nofollow = expression`

Details The value for *expression* must match any of the text that lies between the `<a href="link">` and `</a>` HTML tags. For example, given `<a href="http://myweb/file.htm">term1 term2</a>`, you could use the following:

```
nofollow = term1
```

The link to `http://myweb/file.htm` would not be followed.

You can use wild-card expressions, where the asterisk (`*`) is for text strings and the question mark (`?`) is for single characters.

To be able to also use regular expressions with this option, you must also specify the `regexp` option.

Default K2 Spider follows all links in HTML documents.

norobo

Description Specifies that any `robots.txt` files encountered are ignored.

Type Web crawling only (`gatewaytype = web`)

Syntax `norobo = [true | false]`

Details The `robots.txt` file is used on many Web sites to specify what parts of the site indexers should avoid. The default is to honor any `robots.txt` files.

If you are re-indexing a site and `robots.txt` has changed, the K2 Spider deletes documents that have been newly disallowed by `robots.txt`.

This option must, of course, be used with discretion and extreme care, especially when used in conjunction with `cgio.k`.

Default `robots.txt` files are honored.

See Also `nodocrobo` and <http://info.webcrawler.com/mak/projects/robots/norobots.html>.

pathlen

Description Limits indexing to the specified number of path segments in the URL or file system path.

Syntax `pathlen = num_pathsegments`

Details The path length is determined as follows:

- The host name and drive letter are not included. For example, neither `www.spider.com:80/` nor `C:\` would be included in determining the path length.
- All elements following the host name are included.
- The actual file name, if present, is included. For example, `/world.html` would be included in determining the path length.

Any directory paths between the host and the actual file name are included.

Default 100 path segments.

Example For the following URL, the path length would be 4:

```
http://www.spider:80/comics/fun/funny/world.html
      <-1->  <2> <-3-> <---4--->
```

For the following file system path, the path length would be 3:

```
C:\files\docs\datasheets
      <-1-> <-2-> <---3--->
```


query

Description Specifies a SQL query to determine what to index.

Type ODBC only (`gatewaytype = odbc`)

Syntax `query = "your_WHERE_clause"`

Details Enter only the criterion part of the query, not the entire SQL query. The criterion is the portion of the query that occurs after the "where" clause.

Default K2 Spider will index everything allowed according to the criteria in the `vgwodbc.cfg` configuration file.

Example Assuming you want to index a table, `Table1` with `Col1` as the ODBC gateway primary key (specified as `primarykey:Col1` in `vgwodbc.cfg`), specify the following:

```
query=Col1<1000 AND Col1>100
```

This will be translated by the ODBC gateway into the following SQL query when crawling and gathering document keys:

```
SELECT Col1 FROM Table1 WHERE Col1<1000 AND Col1>100
```

The query option is useful only in the following situations:

- ❑ You are indexing only one repository (DSN) and one table in that repository
- ❑ You use a single `start=` entry with which the query option will be associated

See Also See the documentation for your database server for more information on SQL.

refreshtime

Description Specifies that any documents indexed since the *timeunits* value began are not to be refreshed.

Syntax `refreshtime = timeunits`

Details The syntax for *timeunits* is: *n* day *n* hour *n* min *n* sec, where *n* is a positive integer. Note that there must be spaces, and since the first three letters of each time unit is parsed, you can use the singular or plural form.

Default All documents are refreshed.

Example If you specify `refreshtime = 1 day 6 hour`, then only those documents which were last indexed at least 30 hours and 1 second ago, will be refreshed.

Note This option takes effect after the first time an indexing job is run.

skip

Description Specifies that the K2 Spider is not to index any HTML document that contains the text of *expression* between the starting and ending *HTML_tag*.

Type Web crawling only (*gatewaytype* = web)

Syntax skip = *HTML_tag expression*

Details For multiple *HTML_tag* and *expression* combinations, use multiple instances of the skip option.

You can use wild-card expressions, where the asterisk (*) represents text strings and the question mark (?) represents single characters. For example:

```
skip = a '/my_doc*/year199?'
```

To be able to also use regular expressions with this option, you must also specify the *regexp* option.

Default All HTML documents are indexed.

Example To skip all HTML documents which contain the word "personnel" in the Title element, use the following:

```
skip= title personnel
```

Example To skip all HTML documents which contain both the word "private" and the phrase "internal user" in any paragraph element, use the following:

```
skip = title private  
skip = p "*internal use*"
```

unlimited

Description Specifies that no limits are to be placed on K2 Spider if neither *host* nor *domain* is specified.

Syntax unlimited = [true | false]

Default K2 Spider is limited to the host of the first starting point.

Sample myjob.ini File

Following is the sample `myjob.ini` file installed with K2 Spider. The file was captured as images to fit in this book. The file is broken up into these sections:

- Job Section
- Forms Based Authentication Section
- Crawlers, Indexers, Network and Schedule Sections
- Proxy and DocRouter Sections
- Crawling and Indexing Criteria Section — Examples
- Criterion Section

Job Section

```
; SAMPLE for JOB INI FILE
; General conventions
; 1. Lines starting with ';' or '#' are considered as comments and are ignored
; 2. In-line comment starts with a ';' at any location in a line
; 3. Each section starts with a section name in square brackets
; 4. Attribute name and value must be separated by a '='
; 5. Multiple attributes are specified by repeating the name = value line multiple times
; 6. Attribute names are case-insensitive
; 7. Extra spaces and tabs are allowed and ignored
; 8. Valid attributes values are provided as in-line comments in this file
;*****
;                                JOB SECTION
;start                          specifies a starting URL for an indexing job. You may specify multiple URLs by repeating this line.
;auth                           specifies an authorization file to support authentication for secure path.
;gatewaytype                    specifies the gateway type.
;metamap                        maps customized meta tags to a valid HTTP header.
;mimemap                        an ASCII text file that maps file extensions to MIME-types.
;prefixmap                      an ASCII text file that maps file system paths to web aliases.
;processbif                    a user defined script that processes the bif file before submits it to VDK for indexing.
;loglevel                       specifies types of messages to be logged.
;abspath                        specifies absolute paths for files.
;regex                           specifies the use of regular expressions rather than the default wild-card expressions.
;                                for the following options: -exclude, -indexexclude, -include, -indinclude, -skip, -indskip,
;                                -preferred, and -nofollow.
;casesen                        makes processing case-sensitive by specifying that the spider process separately keys
;                                that differ only in case. Use only for indexing UNIX servers.
;nosubmit                       specifies that K2 spider gathers document locations without indexing them. The documents locations are
;                                stored in BIF, which is not submitted to the collection.
;reparse                        forces parsing of all HTML documents already in the collection.
;nocache                        disables the caching of files during web site indexing.
;maxnumdocs                     maximum number of documents to be downloaded or submitted for indexing.
;maxNumDocsPerHost              maximum number of documents to be downloaded or submitted for indexing per host
;collectlinkinfo                enables collection of link information for 'web' jobs. This
;                                link information is used to improve search relevancy.
;collectanchortext              enables collection of anchor text for 'web' jobs. This
;                                text information is used to improve search relevancy.
;*****
[Job]
start = http://www.verity.com
gatewaytype = web                ;web|filesystems|notes|exchange|odbc|documentum|external
auth =
metamap =
mimemap =
prefixmap=
processbif =
cookiefile =
loglevel = nostdout skip        ;nostdout skip|summary|verbose|debug|trace. The first arg can be omitted.
abspath =
regex =                         ;true|false
casesen =                       ;true|false
nosubmit =                      ;true|false
reparse =                      ;true|false
nocache =                      ;true|false
maxnumdocs =
maxNumDocsPerHost =
collectlinkinfo = false        ;true|false
collectanchortext = false      ;true|false
```

Forms Based Authentication Section

```
;*****
;                                FORMS BASED AUTHENTICATION SECTION
;FormsAuth      specifies details of how forms based authentication is to be done.
;
;  LoginURL identifies the page that collects credentials, Action specifies the
;  page that authenticates the user based on the credentials and ErrorURL identifies
;  the page that is used to display authentication errors to the user. Method allows
;  users to specify whether GET or POST is used for authentication. Charset is the
;  character set of the credentials. Any name/value pairs contained in this section
;  other than those mentioned above are considered to be credentials and included
;  with the authentication request. The ;encrypted comment on any of these credentials
;  indicates that Password is encrypted. You may use k2spider_cli's "encrypt" command
;  to create encrypted passwords. This section can be repeated several times for
;  multiple forms.
;*****
[FormsAuth]
;LoginURL = http://www.mycompany.com/finance/login.asp
;Action = http://www.mycompany.com/finance/submit.asp
;ErrorURL = http://www.mycompany.com/finance/loginError.asp
;Method = GET or POST
;Charset = <CharacterSet>
;Username = <Username>
;Password = <Password> ;encrypted
```

Crawlers, Indexers, Network and Schedule Sections

```
;*****
;                                CRAWLER SPEC SECTION
;crawler spec      specifies the host name and port number for a crawler.
;                  For example: crawler spec = host2:9091. You can list multiple entries.
;*****
[ Crawlers ]
;crawler spec = <crawler spec>
;crawler spec = <crawler spec>
;*****
;                                INDEXER SPEC SECTION
;indexer spec      specifies the host name and port number for an indexer.
;                  For example: indexer spec = host2:9092. You can list multiple entries.
;*****
[ Indexers ]
;indexer spec = <indexer spec>
;indexer spec = <indexer spec>
;*****
;                                NETWORKING SECTION
;agent name        specifies the value for the agent name field that is part of the HTTP request.
;timeout           specifies the time period, in seconds, that K2 spider should wait before timing out on a network
;                  connection and on accessing data.
;retry             specifies the number of times that K2 spider should attempt to access an URL.
;delay             specifies minimum time between HTTP requests in milliseconds.
;header            specifies an HTTP header to be added to the spidering request.
;*****
[ Network ]
agent name = vspider
timeout = 30
retry = 4
delay = 0
header =          ;HTTP header to be added to the spidering request
;*****
;                                SCHEDULING SECTION
;sched mode        if this field is specified as "auto", K2 Spider ignores all other fields in this section.
;frequency         specifies how often the indexing job is to run.
;repeat            specifies the number of minutes the K2 Spider Indexer will wait after the indexing job has finished
;                  before next running that same indexing job.
;*****
[ Schedule ]
sched mode = auto          ;auto|schedule
year =                    ;2000-2037
month =                   ;1-12
day =                    ;1-31
hour =                   ;0-23
minute =                 ;0-59
frequency =              ;daily|weekly|monthly|weekday
repeat =
```

Proxy and DocRouter Sections

```
*****
;
;          HTTP PROXY SECTION
;hostname      host for proxy server
;portnum       port for proxy server
;username       login name for proxy server connections that require authorization to get outside the firewall.
;password       password for proxy server connections that require authorization to get outside the firewall.
;noproxy        specifies host that K2 spider directly access the hosts whose names match those specified.
;*****
[Proxy]
hostname =
portnum =
username =
password =
noproxy =          ;can use the asterisk (*) wild card for text strings, but not the question mark(?) wild card.
                   ;and the -regexp option does not allow you to use regular expressions.
;*****
;          DOCUMENT ROUTING SECTION
;routertype     specifies the router type: Round-Robin, Bound-Crawler-Indexer, Profile-based, Mime-based or Key-Based
;               default is RoundRobin.
;rule           used for key-based, profile-based and mime-based routing.
;               for mime-based and key-based routing, a rule consists of two parts: indexer spec and matching
;               expression (regular or wild card). Documents which match the rule are routed to the specified indexer.
;               for profile-based routing, there are two additional fields after indexer spec and rule expression,
;               these fields are threshold and forceThreshold.
;               refer to documentation for more details on each of these router types.
;rulefile       an ASCII text file that gathers rules.
;*****
[DocRouter]
;routertype = KeyBased          ;RoundRobin | BoundCrawlIndx | KeyBased | ProfileBased | Mimebased
;rule = yamaha:8081 *.verity.com* ;example for key based routing. NOTE: routertype must be set to KeyBased
;rule = yamaha:8082 text/*      ;example for mime based routing. NOTE: routertype must be set to MimeBased
;rule = yamaha:8083 "<stem>Break,News" 70 ;example for profile based routing. NOTE: routertype must be set to ProfileBased
;rulefile =
```

Crawling and Indexing Criteria Section — Examples

```
*****
;          CRAWLING and INDEXING CRITERIA SECTION
;followdup    specifies whether K2 spider follows links within duplicate documents, although only the first
;             instance of any duplicate documents will be indexed.
;followsymlink specifies whether K2 spider follows symbolic links when indexing UNIX file systems.
;maxdocsize   specifies the maximum size, in kilobytes, for documents to be indexed.
;mindocsize   specifies the minimum size, in kilobytes, for documents to be indexed.
;pathlen      limits indexing to the specified number of path segments in the URL or file system path.
;noderobot    specifies ROBOT META tag directives are to be ignored
;norobot      specifies that any robots.txt files encountered are ignored. The default is to honor
;             any robots.txt files.
;unlimited     specifies no limits to be placed on K2 spider. This overwrites -host and -domain.
;jumps        specifies the maximum number of levels deep an indexing job can go from the starting URL.
;cgiok        allows indexing of URLs containing the ? symbol.
;javascriptok enable extraction of hypelinks embedded in JavaScript.
;nodupdetect  disables checksum-based detection of duplicates when indexing web sites.
;detectdupfile enables checksum-based detection of duplicates when indexing file systems.
;ignoreacl    spider should ignore acl change during refresh.
;nofollow     specifies K2 spider cannot follow any links if their description match the expression.
;refreshtime  specifies that any documents which have been indexed since the time units
;             value began are not to be refreshed
;lastmodified specifies that any documents which were modified before the time units value began will not be indexed.
;skip         skip the document if zone description matches the regular expression
;indskip      skip indexing the document if zone description matches the regular expression, but does not affect following
;include      only those files, paths and URLs which match the specified expression(s) will be followed.
;exclude      files, paths and URLs matching the specified expression(s) will not be followed.
;indinclude   specifies that only those files and paths in URLs which match the expressions be followed and indexed.
;indexinclude specifies that the files and paths in URLs which match the expressions are not indexed.
;mimeinclude  specifies MIME types to be included.
;mimeexclude  specifies MIME types which are neither followed nor indexed.
;indmimeinclude specifies that only those MIME types which match the expressions be followed and indexed.
;indmimeexclude specifies that only those MIME types which match the expressions be followed but not indexed.
;domain       limits indexing to the specified domain(s).
;host         limits indexing to the specified host(s).
;query        specifies a SQL query to determine what to index for odbc.
```


Criterion Section

```
*****
[Criterion]
followdup =                               ;true|false
followsymlink =                           ;true|false
maxdocsize = 20000                         ;kbytes
mindocsize = 0                             ;kbytes
pathlen = 100
nodocrobo =                               ;true|false
norobo =                                   ;true|false
unlimited =                                ;true|false
jumps = 12                                 ;0-254
cgick = false                              ;true|false
javascriptok = false                       ;true|false
nodupdetect =                              ;true|false
detectdupfile =                            ;true|false
ignoreacl =                               ;true|false
nofollow =                                ;regular expression or wild card expression
refreshtime =                               ;<num> day|hour|min|sec
lastmodified =                             ;<num> day|hour|min|sec
skip=                                       ;arg1: zone name, arg2: regular expression or wild card expression
indskip =                                  ;arg1: zone name, arg2: regular expression or wild card expression
include = *                                ;regular expression or wild card expression
exclude =                                  ;regular expression or wild card expression
indinclude = *                             ;regular expression or wild card expression
indexexclude =                             ;regular expression or wild card expression
mimeinclude = text/*                       ;regular expression (with option specified), or wild card expression
mimeexclude =                             ;regular expression (with -regexp option specified), or wild card expression
indmimeinclude = text/*                   ;mimetype expression
indmimeexclude =                           ;mimetype expression
domain = verity.com                       ;text strings
host =                                     ;text strings
query =                                    ;text string
```

Evaluating “include” and “exclude” Criteria

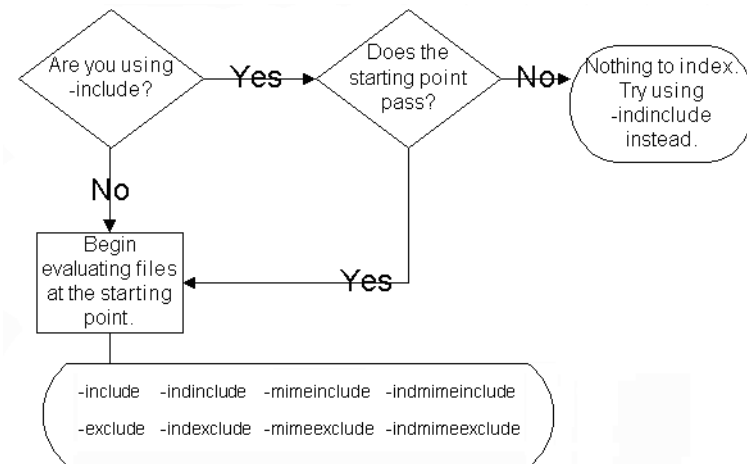
When you specify any of the include or exclude criteria with a K2 Spider indexing job, all candidates (URL, path and file) are evaluated against all of the specified criteria. The evaluation occurs as a logical AND across the criteria options.

Evaluation Workflow

The basic workflow for evaluating criteria is illustrated in the following diagram.

Note K2 Spider evaluates each criteria option in turn, one at a time, and the order is arbitrary.

Figure 9-2 Workflow of K2 Spider evaluating criteria



The include option concerns following and indexing. When you use it in your indexing job, the starting point must be passed to determine whether or not K2 Spider can proceed to any of the files past the starting point.

For example, if you specify `include = '*memo*'` and the value for start in the job INI file is as follows, `start = http://web.verity.com/docs`, nothing will happen. The starting point does not match the include string.

Since `indinclude` only concerns indexing while always making it possible to follow, you should use it instead of `include` in most cases.

Sample Indexing Job

The following example indexing job illustrates how candidates are evaluated against the criteria.

K2 Spider job.ini Entries

In this example, the following values are specified in the job INI file.

```
[Job]
start = http://web.verity.com/intdocs

[Criterion]
jumps = 2
domain = verity.com
indinclude = *acct*
indexexclude = *memo*
indmimeinclude = apps/msword
indmimeexclude = text/html
```

Note The indinclude option is being used because using include would result in zero files indexed. The reason is that the starting point would not qualify according to the include string. Therefore, the indinclude option must be used instead, as it will allow following from the starting point while only indexing what matches.

Candidates from Starting Point

Based on the starting point in the example, assume the following files are available:

In this directory...	These files are available for indexing...
/intdocs/	index.html, which contains links to files in a subdirectory, accounting.
/intdocs/accounting/	acctmemo.doc, acctclose.doc, acctmemo2.doc and acct1.htm, acct2.htm. Both HTML files link to Microsoft Word documents, acct1.doc and acct2.doc.

Since the example is not using include, you can assume that the starting point is valid. Start evaluating with index.html in the /intdocs/ directory.

Evaluating the Candidates

The URL `http://web.verity.com/intdocs` is processed, and the file `index.html` is retrieved because the Web server is configured to server `index.html` for the virtual directory `/intdocs`.

Processing `index.html`

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `index.html` fails. At this point, `index.html` will not be included for indexing. On to the next criteria.
2. `indexexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `index.html` does not contain those characters so it passes. However, the initial criteria still holds meaning at this point, `index.html` is still not to be included for indexing. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `index.html` fails this criteria. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `index.html` fails.
5. The conclusion is `index.html` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `index.html` is examined for links to other files and other links.

Following is a simplistic representation of a part of `index.html`, to illustrate the links and documents being evaluated in this example.

```
<body>
<a href="accounting/acctmemo.doc">Memo Month 1</a>
<a href="accounting/acctmemo2.doc">Memo Month 2</a>
<a href="accounting/acctclose.doc">Accounts Closed</a>
<a href="accounting/acct1.htm">First half of month</a>
<a href="accounting/acct2.htm">Second half of month</a>
</body>
```

Each of the files are then processed in turn, one at a time.

Processing acctmemo.doc

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctmemo.doc` passes. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctmemo.doc` fails. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acctmemo.doc` passes this criteria. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctmemo.doc` passes.
5. The conclusion is `acctmemo.doc` cannot be included for indexing. Although it passes three of the criteria, it cannot be indexed because all criteria must be passed.

Processing acctmemo2.doc

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctmemo2.doc` passes. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctmemo2.doc` fails. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acctmemo2.doc` passes this criteria. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctmemo2.doc` passes.
5. The conclusion is `acctmemo2.doc` cannot be included for indexing. Although it passes three of the criteria, it cannot be indexed because all criteria must be passed.

Processing acctclose.doc

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acctclose.doc` passes. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acctclose.doc` passes. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acctclose.doc` passes. On to the next criteria.

4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acctclose.doc` passes.
5. The conclusion is `acctclose.doc` is to be included for indexing.

Processing `acct1.htm`

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct1.htm` passes. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct1.htm` passes. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acct1.htm` fails. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct1.htm` fails.
5. The conclusion is `acct1.htm` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `acct1.htm` is examined for links to other files and other links.

Following is a simplistic representation of a part of `acct1.htm`, to illustrate the links and documents being evaluated in this example.

```
<body>
<a href="accounting/acct1.doc">Accounts 1st month</a>
</body>
```

The file `acct1.doc` will be processed after `acct2.htm`.

Processing `acct2.htm`

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct2.htm` passes. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct2.htm` passes. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acct2.htm` fails. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct2.htm` fails.

5. The conclusion is `acct2.htm` is not to be included for indexing. However, since there are no criteria preventing it from being parsed, `acct1.htm` is examined for links to other files and other links.

Following is a simplistic representation of a part of `acct2.htm`, to illustrate the links and documents being evaluated in this example.

```
<body>
<a href="accounting/acct2.doc">Accounts 2nd month</a>
</body>
```

The file `acct2.doc` is then processed with `acct1.doc`.

Processing `acct1.doc` and `acct2.doc`

1. `indinclude = *acct*` — This criteria states that only files that contain the characters `acct` will be included for indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
2. `indexclude = *memo*` — This criteria states that any file that contains the characters `memo` will be excluded from indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
3. `indmimeinclude = apps/msword` — This criteria states that only files that are of the MIME Type `apps/msword` are to be included for indexing. `acct1.doc` and `acct2.doc` pass. On to the next criteria.
4. `indmimeexclude = text/html` — This criteria states that any file that is of the MIME Type `text/html` is to be excluded from indexing. `acct1.doc` and `acct2.doc` pass.
5. The conclusion is `acct1.doc` and `acct2.doc` are to be included for indexing.

NOTES

The following table summarizes what happened with each file.

Candidate	-indinclude *acct*	-indexclude *memo*	-indmimeinclude apps/msword	-indmimeexclude text/html	Conclusion
index.html	Fails	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acctmemo.doc	Passes	Fails	Passes	Passes	Not indexed.
acctclose.doc	Passes	Passes	Passes	Passes	Indexed.
acctmemo2.doc	Passes	Fails	Passes	Passes	Not indexed.
acct1.htm	Passes	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acct2.htm	Passes	Passes	Fails, but parsed	Fails	Parsed but not indexed.
acct1.doc	Passes	Passes	Passes	Passes	Indexed.
acct2.doc	Passes	Passes	Passes	Passes	Indexed.

- The ordering of the criteria across the table matches the ordering of the criteria in the `job.ini` example. Although K2 Spider evaluates each criteria option in turn, one at a time, the order is arbitrary.
- The linear nature of processing these files is a product of documenting them. The files can actually be processed simultaneously depending on the number of connections.
- By using the “ind” criteria, the K2 Spider is able to follow links to the limit of jumps, while still only indexing those files which meet the desired criteria.
- It is technically not necessary to also use `-indmimeexclude`. Using `indmimeinclude` implies that you are excluding everything else.
- It is necessary to have both `indinclude = *acct*` and `indexclude = *memo*` because you do not want to index any files which contain both. In most cases, however, using an include value implies excluding everything else, and vice versa.
- When you find that you are not indexing as many files as you think you should, review your indexing criteria to ensure you are not being too restrictive.

The Use of Last-Modified Date

When Web crawling to gather HTML documents for indexing, K2 Spider looks for the date the document was last modified, in the form of a field named Last-Modified. The value of Last-Modified is used to determine if documents should be indexed again.

How Last-Modified is Used

The types of documents encountered during indexing are as follows:

- [New Documents](#)
- [Previously Indexed Documents](#)
- [Dynamic Documents](#)

How the last-modified value is used depends on the type of document.

New Documents

For HTML documents that have never been indexed, the value for Last-Modified is not compared to anything, as nothing exists in the persistent store. Instead, the Last-Modified date, if it exists, is stored in the `last_modified_date` field of the persistent store for the collection into which the document is being indexed.

Previously Indexed Documents

For HTML documents which have been indexed, and for which a value exists in the `last_modified_date` field in the persistent store, K2 Spider compares the retrieved document's Last-Modified value with `last_modified_date`. What happens to the document depends on the outcome of the comparison.

- If no value exists in the persistent store, or it is older than the current value, then the document is indexed into the collection and the `last_modified_date` field is updated in the persistent store.
- If a value exists in the persistent store and it is the same or newer than the current value provided by the retrieved document, then the document is not indexed.
- If a Last-Modified value is not provided with a retrieved document, then K2 Spider treats the document as new and always indexes it.

Dynamic Documents

If you are dealing with dynamically-generated HTML documents, then there may never be a Last-Modified date and so the document may always be indexed. A workaround is to incorporate a meta tag into the processing of the dynamic documents and take advantage of the `metamap` option.

K2 Spider Examples

This chapter contains indexing examples with specific criteria, and some in-depth examples covering specific situations.

This chapter includes the following sections:

- [Examples](#)
- [Specific Situations and Concepts](#)

Examples

This section contains examples for running K2 Spider indexing jobs with the Lotus Notes Gateway, the ODBC Gateway and the Documentum Gateway.

This section includes:

- [Indexing Workflow](#)
- [Indexing a Notes Repository](#)
- [Indexing an ODBC Repository](#)
- [Indexing a Documentum Repository](#)

Indexing Workflow

In broad terms, indexing with K2 Spider consists of the following:

1. Set up your environment and make any other preparations that are necessary depending on the repository you are indexing.

The particular preparations are described in each of the repository-specific sections that follow.

2. Run the `k2spider_srv` processes) to establish your Controller, Crawler and Indexer.

Note This step can be as simple or as complex as your needs warrant. For the purposes of these examples, a two-stage environment will be used where one `k2spider_srv` process runs as the Controller and another `k2spider_srv` process runs as both the Crawler and Indexer.

3. Create a job initialization file that contains the parameters for your indexing job. These parameters include which Crawler and Indexer will execute the job, what types of constraints there are on gathering and indexing, and when the job should run.
4. Start a K2 Spider Client session with `k2spider_cli`.

Since K2 Spider is a client/server system, you use the K2 Spider Client to interact with the Controller to create and start the indexing job, and get information about how the job is doing.

Indexing a Notes Repository

To index a Notes repository, follow these steps:

1. Ensure that the indexing host has access to the Notes repository and that the environment is properly configured. This is the host on which the K2 Spider Server (`k2spider_srv`) processes are running.

On Windows, preparations include the following:

- ❑ There is a Notes client or a Domino server installed on the indexing host. The application path must be in your system environment path.
- ❑ The Notes ID on the indexing host has the necessary privileges to access all Notes databases you want to index.
- ❑ If the Notes ID is password protected, it needs to be initialized and shared with other applications. See your Notes documentation for more information.

On Unix, preparations include the following:

- ❑ There is a Domino server installed on the indexing host.
- ❑ The Notes ID on the indexing host has the necessary privileges to access all Notes databases you want to index.
- ❑ If the Notes ID is password protected, it needs to be initialized and shared with other applications. See your Notes documentation for more information.
- ❑ The following environment variables must be set prior to starting any Verity programs:

```
setenv LOTUSDIR /opt/lotus
setenv NOTES_DATA_DIR /opt/lotus/data
setenv Notes_ExecDirectory /opt/lotus/notes/latest/sunspa

setenv LD_LIBRARY_PATH /opt/lotus/notes/latest/
sunspa:$LD_LIBRARY_PATH
setenv PATH /opt/lotus/data:/opt/lotus/notes/latest/sunspa/res/
C:$PATH
```

Note This information pertains to Solaris. Information for AIX will be different.

2. Specify what you will be indexing by creating a style set with the StyleSet Editor. For information on using the StyleSet Editor to create your style set, see the *Verity K2 Lotus Notes Gateway Guide*.

3. Start the K2 Spider Controller process:

```
k2spider_srv -controller -port 7000 -workpath mypath/workdir
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

4. Start the K2 Spider Crawler/Indexer process.

```
k2spider_srv -crawler localhost:7000  
-indexer localhost:7000 -port 7001  
-workpath mypath/ciworkdir  
-gateway notes  
-style mypath/style  
-collection mypath/notes1coll  
-gwauth mypath/style/vgwnotes.auth
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

The command is broken up for better readability. You must execute the entire command as a continuous string.

The `vgwnotes.auth` file contains the necessary credentials for indexing. If you use the StyleSet Editor, this file is created for you. Make sure you specify the correct path. For information on using the StyleSet Editor to create your style set, see the *Verity K2 Lotus Notes Gateway Guide*.

5. Create a job INI file for the indexing job. Following is a sample file:

```
[Job]  
jobname = notesjob  
jobworkpath = mypath/workdir  
gatewaytype = notes  
loglevel = nostdout verbose  
[Crawlers]  
crawlerSpec = localhost:7001  
[Indexers]  
indexerSpec = localhost:7001
```

Note The above job INI file may not represent the complete set of options required for your job. See [Chapter 9, “Job Initialization File Reference”](#) for more information. Your path information and the values for `crawlerSpec` and `indexerSpec` will differ.

The key options in this job INI file are as follows:

- ❑ `jobname` — The name you provide here will be what you use when you interact with the Controller using K2 Spider Client.
- ❑ `gatewaytype` — The value for this example must be `notes` because you are using the Lotus Notes Gateway to index a Notes repository (Domino server).
- ❑ `indexerSpec` and `crawlerSpec` — Note that the port number is the same for both. This is because both services run in the same `k2spider_srv` process. In step 3, the port specified for the Crawler/Indexer process is 8001, the port specified for both `indexerSpec` and `crawlerSpec` here in the job INI file.

Note There is no need to specify a blank `start` entry if you do not need extra filtering. The starting points are included in the Lotus Notes Gateway configuration file.

6. Start the K2 Spider Client and create the job as illustrated in the following sequence:

```
k2spider_cli -controller localhost -port 7000
K2S Client> create mypath/notesjob.ini
(ind040001) Sent job to crawler and indexer successfully.
K2S Client>
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 8, “Using K2 Spider Client”](#) for more information.

7. Check the status of the job.

While you are still in a K2 Spider Client session from step 5, you can check the status of the job using the `getstat` command as follows.

```
k2spider_cli>getstat notesjob
Job Name = notesjob
Job State = Finished
Failed to fetch keys = 8
Crawled keys = 1806
Skipped keys = 200
Indexed keys = 1606
Deleted keys = 0
Bad keys = 0
Execution time: 2 minutes 57 seconds
k2spider_cli>
```

At this point you have created a collection and indexed content into that collection.

Indexing an ODBC Repository

To index an ODBC repository, follow these steps:

1. Ensure that the environment is properly configured on the indexing host. This is the host on which the K2 Spider Server (`k2spider_srv`) processes are running.

On Windows, preparations include the following:

- ❑ A DSN is properly configured.
- ❑ The system path includes the path to the relevant ODBC drivers.

On UNIX, preparations include the following:

- ❑ The `ODBCINI` environment variable refers to the correct `odbc.ini` file.
- ❑ The `PATH` and `LD_LIBRARY_PATH` variables are configured. `LD_LIBRARY_PATH` must include the path to the relevant ODBC drivers.

2. Specify what you will be indexing by creating a style set with the StyleSet Editor. For information on using the StyleSet Editor to create your style set, see the *Verity K2 ODBC Gateway Guide*.

3. Start the K2 Spider Controller process:

```
k2spider_srv -controller -port 8000 -workpath mypath2/workdir
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

4. Start the K2 Spider Crawler/Indexer process.

```
k2spider_srv -crawler localhost:8000  
-indexer localhost:8000  
-port 8001  
-workpath mypath2/ciworkdir  
-gateway odbc  
-style mypath2/style  
-collection mypath2/odbccoll1  
-gwauth mypath2/style/vgwodbc.auth
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

The command is broken up for better readability. You must execute the entire command as a continuous string.

The `vgwodbc.auth` file contains the necessary credentials for indexing. If you use the StyleSet Editor, this file is created for you. Make sure you specify the correct path. For information on using the StyleSet Editor to create your style set, see the *Verity K2 ODBC Gateway Guide*.

5. Create a job INI file for the indexing job. Following is a sample file:

```
[Job]
jobname = odbcjob
jobworkpath = mypath2/workdir
gatewaytype = odbc
loglevel = nostdout verbose
[Crawlers]
crawlerSpec = localhost:8001
[Indexers]
indexerSpec = localhost:8001
```

Note The above job INI file may not represent the complete set of options required for your job. See [Chapter 9, “Job Initialization File Reference”](#) for more information. Your path information and the values for `crawlerSpec` and `indexerSpec` will differ.

The key options in this job INI file are as follows:

- `jobname` — The name you provide here will be what you use when you interact with the Controller using K2 Spider Client.
- `gatewaytype` — The value for this example must be `odbc` because you are using the ODBC Gateway to index an ODBC repository.
- `indexerSpec` and `crawlerSpec` — Note that the port number is the same for both. This is because both services run in the same `k2spider_srv` process. In step 3, the port specified for the Crawler/Indexer process is 8001, the port specified for both `indexerSpec` and `crawlerSpec` here in the job INI file.

Note There is no need to specify a blank start entry if you do not need extra filtering. The starting points are included in the ODBC Gateway configuration file.

6. Start the K2 Spider Client and create the job as illustrated in the following sequence:

```
k2spider_cli -controller localhost -port 8000
K2S Client> create mypath2/odbcjob.ini
(ind040001) Sent job to crawler and indexer successfully.
K2S Client>
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 8, “Using K2 Spider Client”](#) for more information.

7. Check the status of the job.

While you are still in a K2 Spider Client session from step 5, you can check the status of the job using the `getstat` command as follows.

```
k2spider_cli>getstat odbcjob
Job Name = odbcjob
Job State = Finished
Failed to fetch keys = 2
Crawled keys = 1000
Skipped keys = 100
Indexed keys = 900
Deleted keys = 0
Bad keys = 0
Execution time: 12 minutes 19seconds
k2spider_cli>
```

At this point you have created a collection and indexed content into that collection.

Indexing a Documentum Repository

To index a Documentum repository, follow these steps:

1. Ensure that the environment is properly configured on the indexing host. This is the host on which the K2 Spider Server (`k2spider_srv`) processes are running.

On Windows, preparations include the following:

- The `DMCL_CONFIG` environment variable must include the path to the appropriate `dmcl.ini` file that contains a valid Documentum broker host name entry.

On UNIX, preparations include the following:

- The `LD_LIBRARY_PATH` variable contains the following directory:

`/path/verity/k2/platform/bin`

where *path* is the installation directory and *platform* is relevant to your operating system. For example, for Solaris, *platform* is `_ssol26`.

Note You may need to manually update `LD_LIBRARY_PATH`.

- The `DMCL_CONFIG` environment variable contains the following entry:

`/path/verity/k2/platform/bin/dmcl.ini`

where *path* is the installation directory and *platform* is relevant to your operating system. For example, for Solaris, *platform* is `_ssol26`.

You specify the appropriate doc broker host name in `dmcl.ini`.

2. Specify what you will be indexing by creating a style set with the StyleSet Editor. For information on using the StyleSet Editor to create your style set, see the *Verity K2 Documentum Gateway Guide*.
3. Start the K2 Spider Controller process:

```
k2spider_srv -controller -port 9000 -workpath mypath3/workdir
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

4. Start the K2 Spider Crawler/Indexer process.

```
k2spider_srv -crawler localhost:9000  
-indexer localhost:9000  
-port 9001  
-workpath mypath3/ciworkdir  
-gateway documentum  
-style mypath3/style  
-collection mypath3/dctmcoll1  
-gwauth mypath3/style/vgwdctm.auth
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 7, “Running K2 Spider Server”](#) for more information.

The command is broken up for better readability. You must execute the entire command as a continuous string.

The `vgwdctm.auth` file contains the necessary credentials for indexing. If you use the StyleSet Editor, this file is created for you. Make sure you specify the correct path. For information on using the StyleSet Editor to create your style set, see the *Verity Documentum Gateway Guide*.

5. Create a job INI file for the indexing job. Following is a sample file:

```
[Job]  
jobname = dctmjob  
jobworkpath = mypath3/workdir  
gatewaytype = documentum  
loglevel = nostdout trace  
[Crawlers]  
crawlerSpec = localhost:9001  
[Indexers]  
indexerSpec = localhost:9001
```

Note The above job INI file may not represent the complete set of options required for your job. See [Chapter 9, “Job Initialization File Reference”](#) for more information. Your path information and the values for `crawlerSpec` and `indexerSpec` will differ.

The key options in this job INI file are as follows:

- `jobname` — The name you provide here will be what you use when you interact with the Controller using K2 Spider Client.

- `gatewaytype` — The value for this example must be `documentum` because you are using the Documentum Gateway to index a Documentum repository.
- `indexerSpec` and `crawlerSpec` — Note that the port number is the same for both. This is because both services run in the same `k2spider_srv` process. In step 3, the port specified for the Crawler/Indexer process is 8001, the port specified for both `indexerSpec` and `crawlerSpec` here in the job INI file.

Note There is no need to specify a blank `start` entry if you do not need extra filtering. The starting points are included in the Documentum Gateway configuration file.

6. Start the K2 Spider Client and create the job as illustrated in the following sequence:

```
k2spider_cli -controller localhost -port 9000
K2S Client> create mypath3/dctmjob.ini
(ind040001) Sent job to crawler and indexer successfully.
K2S Client>
```

Note The above command may not represent the complete set of options required for your job. See [Chapter 8, “Using K2 Spider Client”](#) for more information.

7. Check the status of the job.

While you are still in a K2 Spider Client session from step 5, you can check the status of the job using the `getstat` command as follows.

```
k2spider_cli>getstat dctmjob
Job Name = dctmjob
Job State = Finished
Failed to fetch keys = 2
Crawled keys = 18
Skipped keys = 0
Indexed keys = 18
Deleted keys = 0
Bad keys = 0
Execution time: 5 minutes 7 seconds
k2spider_cli>
```

At this point you have created a collection and indexed content into that collection.

Specific Situations and Concepts

The examples in this section cover specific situations and concepts.

- Customizing the Last-Modified Date
- Indexing Secure Web Sites
- Indexing with Proxy Servers
- Indexing Network, UNC and Secure Paths on Windows
- Indexing Network, UNC and Secure Paths on UNIX
- Defining the Authenticating Repository for Secure File System Collections
- Access Control List (ACL) Caching — Domain versus Local Users
- Prefix Mapping
- Setting MIME Types
- Using StyleSet Editor in Standalone Mode

Customizing the Last-Modified Date

There may be cases in which you want to customize the Last-Modified date value.

Scenarios for Customizing

There are two scenarios under which you would want to customize the Last-Modified date value. These are:

- The Last-Modified meta tag is missing, requiring you to add one.

There are two possible reasons for the Last-Modified meta tag to be missing. These are:

- The documents are created dynamically (on-the-fly).
- The Web server is configured to omit the value.

Under this scenario, you will not be able to use the Last-Modified meta tag, or any variation thereof. In order to create a custom value, or override an existing value, the Last-Modified meta tag must exist in the HTML document itself as it is retrieved from the web server. If you do not have any control over the web server or documents in question, you can still exercise some control over when documents are indexed by using `refreshtime` with inclusion or exclusion criteria.

- The value provided does not contain a satisfactory value, and you want to override it. Note that you can do this for file system indexing as well as Web site indexing.

Whether you need to insert a value for Last-Modified date, or you want to override an existing value, the steps are the same.

How to Customize the Last-Modified Date Value

Review the steps below for a description of how to customize the Last-Modified date value.

Step 1 — Add a Meta Tag

The first step is to ensure that your HTML documents contain a meta tag which can be used for the Last-Modified date. If your documents do not contain such a meta tag, you must add one.

Meta tags use the syntax:

```
<meta name="name" content="content">
```

where *name* is any string which you will later use in the text map file, and *content*, is the date and time you want to use for Last-Modified. The value for *content* must be in one of the following date formats:

Date format	Example
RFC822 (updated by RFC 1123)	Example: Sun, 06 Nov 1999 08:49:37 GMT
RFC850 (obsoleted by RFC 1036)	Example: Sunday, 06-Nov-99 08:49:37 GMT
ANSI C's asctime() format	Example: Sun Nov 6 08:49:37 1999
Warning! The day value must occupy two spaces.If you only have one digit, you must provide an extra space between the month and the digit.	

For dynamic documents, such as database middleware or scripts, you must find a way to incorporate the meta tag into the processing of the document.

Step 2 — Create a Text File

Since K2 Spider by default only recognizes the valid HTTP header Last-Modified along with its value, you will need to map your custom meta tag name to the expected name Last-Modified.

In a text editor, create a simple text file with a single entry for mapping your custom meta tag name to the expected name, Last-Modified. The syntax for the entry is:

```
name Last-Modified Y/N
```

where *name* is the meta tag name you specified in step 1, and *Y/N* is an override flag which can be either “Y” for yes or “N” for no.

Flag	Description
Y	When you use the “Y” flag, the value for the custom meta tag overrides the value for Last-Modified, even if both values are present and differ.
N	When you use the “N” flag, the value for the custom meta tag will be used only if there is no value for Last-Modified. If a value for Last-Modified exists, then that is given precedence.

Step 3 — Run a K2 Spider indexing job with the metamap option

Once you have your custom meta tag name in the HTML documents and you have created a text file in which you map this custom meta tag name to the expected meta tag name Last-Modified, you can run a K2 Spider indexing task using the `metamap` option in the job INI file.

For example:

```
[Job]  
metamap = path/mapmeta.txt
```

Remember that the simple text file `mapmeta.txt` contains the mapping for your custom meta tag to the Last-Modified meta tag.

Indexing Secure Web Sites

Some sites secure documents by requiring authentication on the realm in which they are stored. This is typically accomplished using the administration functions of the Web server. In order to retrieve documents for indexing from such secured directories, the K2 Spider can use the `auth` option in the `[Job]` section of the `job.ini` file to reference an authorization file. The authorization file is a simple ASCII text file that you must create.

K2 Spider automatically supports Basic, NTLM and Public authentication with Microsoft's Internet Information Server Web server. These authentication mechanisms are also automatically supported when Verity needs to retrieve documents for viewing.

Procedure

Indexing web sites secured by realms involves the following tasks:

1. Prepare an authorization file for later use with the `auth` option in the `job.ini` file.

The authorization file is an ASCII text file that you must create manually in a text editor. The authorization file contains one record per line. Each line consists of the following columns, separated by white space.

Server Name	URL Path	Username	Password

Here is a sample authorization file:

#Server Name	URL Path	Username	Password
cairo.myco.com:8080	/docs	Aladdin	open
www.123456.com	/arts/columns	barelto	94089

For the sake of simplicity, you can save the authorization file in the `/style` directory of the collection into which you are indexing, and use a name such as `vgwhttp.auth`.

If you have a large number of web sites that share the same credentials, you can specify the following in your authorization file:

#Server Name	URL Path	Username	Password
*	" "	Aladdin	open

This allows K2 Spider to use the credentials `Aladdin` and `open` for all web sites specified in the indexing job for which you are using this authorization file.

Note To use encrypted passwords, use the K2 Spider Client `encrypt` option. For more information, see [“encrypt” on page 214](#).

2. Specify the `auth` option, with the appropriate authorization file, in the `job.ini` file.

For example:

```
[Job]
start = http://host1.domain.com
auth = C:\colls\coll1\style\vgwhttp.auth
```

Note You must include a domain name in the URL for your starting points.

3. Create a `vgwhttp.cfg` file in which you define the secured realms so documents can be retrieved based on results list filtering and for viewing.

For more information, see [“The vgwhttp.cfg File” on page 340](#).

4. Run an indexing job.
5. Search the collection and view the documents retrieved from the secured web sites.

The vgwhttp.cfg File

When you specify the `auth` option in a `job.ini` file for an indexing job, you are only configuring the ability for K2 Spider to gather documents to be indexed into a collection. To support results list filtering and viewing, you must specify the appropriate information in an HTTP gateway configuration file, called `vgwhttp.cfg`.

Note There may already be a `vgwhttp.cfg` file in your style set. These instructions assume you are creating one from scratch. If one already exists, just copy the relevant information described in these instructions into the appropriate section of the existing `vgwhttp.cfg` file.

1. In a text editor, open a new, blank document and type the following information:

```
# vgwhttp.cfg - HTTP gateway configuration file
$control:1
```

Note If you are editing an existing `vgwhttp.cfg` file, due to adding more starting points to an existing collection for example, you can proceed to step 3 and just add more repository sections as necessary.

If you are also specifying information in the `[Proxy]` section of the `job.ini` file for your indexing job, you must also specify information relevant to those options in the

`vgwhttp.cfg` file. For more information, see [“Indexing with Proxy Servers” on page 344](#).

You would add the proxy-related information after the closing curly bracket `}` of the last `repository` section, and before the end-of-file designation of `$$`.

2. Save the text file as `vgwhttp.cfg` in the `/style` directory for the collection associated with the indexing job in which you specified the `auth` option in a `job.ini` file.
3. In your `vgwhttp.cfg` file, add a `repository` section for each group of URLs for which the same credentials apply.

For example, in your authorization file, you might have the same credentials (`user1`, `password1`) for several different realms. You can group the URLs that correspond to those realms into a single `repository` section.

repository Section Syntax

The syntax for a `repository` section is as follows. You can specify multiple `repository` sections.

```
repository: name
{
    securityModuleId: mod_id
    url: protocol://host:port/path/*.
    loginurl: protocol://host:port/path/
}
```

name — The repository name is an arbitrary label used to identify the URL or group of URLs that share a set of credentials. Use any unique alphanumeric string that enables you to easily identify this `repository` section.

mod_id — The *mod_id* defines whether the security module is unsecured or secured. Valid values are:

`0x0` — Use this value if you do not want results filtered based on user. In other words, you would allow a user to see all possible results, while still restricting document access at view time.

`0xdff4` — Use this value if you want to filter results so that a user cannot even see results unless he or she has the proper access.

protocol://host:port/path/.* — Specify the *protocol* (`http` or `https`), host name (*host*), port number (*port*), and path (*path*) that correspond to the realm that requires authentication.

For example, a web server may define the realm `NeedToKnow` as the path:

`http://cairo.myco.com:8000/restricted/docs`

You would then specify the following for *protocol://host:port/path/*.*:

```
http://cairo.myco.com:8000/restricted/docs/.*
```

`loginurl: protocol://host:port/path` — The `loginurl` entry enables you to define a path against which you can test a user's credentials. If the credentials fail on the value for `loginurl`, then the gateway will prompt the user for the proper credentials if the search application enables it. Otherwise, no results are returned to that user.

Note You can only specify the specific URL path. Do not include any wildcard expressions.

For example, if you have several realm URLs for the server `cairo.myco.com:8000`, such as

```
http://cairo.myco.com:8000/data/fyio
```

and

```
http://cairo.myco.com:8000/otherdata/fyio
```

you can define a `loginurl` entry as:

```
loginurl: http://cairo.myco.com:8000
```

This feature helps to improve the performance of the gateway when results list filtering is implemented.

Sample vgwhhttp.cfg File

Following is a sample vgwhhttp.cfg file based on this information:

- Realm 1 is defined as: http://cairo.myco.com:8000/data/fyio
- Realm 2 is defined as: http://cairo2.myco.com:8008/otherdata/fyio
- You want to filter results based on credentials

Here is the vgwhhttp.cfg file:

```
# vgwhhttp.cfg - HTTP gateway configuration file
$control:1
repository: data
{
    securityModuleId: 0xdff4
    url:http://cairo.myco.com:8000/data/fyio/*.
    loginurl: http://cairo.myco.com:8000/data/fyio
}
repository: otherdata
{
    securityModuleId: 0xdff4
    url:http://cairo2.myco.com:8008/otherdata/fyio/*.
    loginurl: http://cairo2.myco.com:8008/otherdata/fyio
}
$$
```

Indexing with Proxy Servers

Some web sites cannot be accessed without going through a proxy server, which may also be secured and require authentication. In order to retrieve documents for indexing from such web sites, K2 Spider uses the information in the `[Proxy]` section of the `job.ini` file for an indexing job.

If there are any web sites that can be accessed directly, without having to go through the proxy server, then you can use the `noproxy` option to specify them and include them in the same indexing job.

Procedure

Indexing web sites secured behind a proxy server involves the following tasks:

1. Create a `vgwhttp.cfg` file.

In the `vgwhttp.cfg` file, you will specify the information you will later use in the `[Proxy]` section of the `job.ini` file so that documents can be retrieved for results list viewing.

For more information, see [“The vgwhttp.cfg File” on page 345](#).

2. Run an indexing job and specify in the `[Proxy]` section of the `job.ini` file the same information you specified in the `vgwhttp.cfg` file.

For example:

```
[Proxy]
hostname = proxyhost
portnum = port
username = username
password = password
noproxy = host_1
noproxy = host_N
```

3. Search the collection and view the documents retrieved from the web sites.

The vgwhhttp.cfg File

When you specify information in the [Proxy] section of a job.ini file for an indexing job, you are only configuring the ability for K2 Spider to gather documents to be indexed into a collection. To support results list viewing, you must specify the appropriate information in an HTTP gateway configuration file, called vgwhhttp.cfg.

Note There may already be a vgwhhttp.cfg file in your style set. These instructions assume you are creating one from scratch. If one already exists, just copy the relevant information described in these instructions into the appropriate section of the existing vgwhhttp.cfg file.

1. In a text editor, open a new, blank document and type the following information:

```
# vgwhhttp.cfg - HTTP gateway configuration file
$control:1
```

Note If you are editing an existing vgwhhttp.cfg file, due to adding more starting points to an existing collection for example, you can proceed to step 3 and just add more proxy sections as necessary.

If you are also using the auth option in your indexing job's job.ini file, you must also specify information relevant to that option in the vgwhhttp.cfg file. For more information, see ["Indexing Secure Web Sites" on page 339](#).

You would add the authorization-related information after the closing curly bracket } of the proxy section, and before the end-of-file designation of \$\$.

2. Save the text file as vgwhhttp.cfg in the /style directory for the collection associated with the indexing job in which you will be specifying the [Proxy] section information.
3. In your vgwhhttp.cfg file, add a proxy section with the relevant information.
The syntax for a proxy section is described in the next section,

proxy Section Syntax

The syntax for a proxy section is as follows:

```
proxy: hostname portnum
{
    proxyAuth: username password
    noproxy:
    {
        server: hostname_or_IP_address
    }
}
```

hostname portnum — The host name and port number of the proxy server. These correspond to entries of the same name (*hostname* and *portnum*) in a `job.ini` file in the `[Proxy]` section.

username password — The credentials required for a secured proxy server. These correspond to entries of the same name (*username* and *password*) in a `job.ini` file in the `[Proxy]` section.

WARNING! The value for password must be in plain text.

This information is optional. Leave the `proxyAuth` line out if your proxy server does not require credentials.

hostname_or_IP_address — The host name or IP address of web servers that can be accessed directly from the computer on which `k2spider_srv` is running without having to go through the specified proxy server.

Note Do not specify a port number; only the host name or IP address of a web server.

You can specify up to 255 entries for `server:hostname_or_IP_address`. You can use the question mark (?) and asterisk (*) wildcard characters to define the *hostname_or_IP_address*, where ? represents a single character, and * represents a string of characters.

The `noproxy` entry and the `server` sub-entry are optional. If you do not need to define any directly accessible hosts, you can omit the following lines:

```
noproxy:
{
    server: hostname_or_IP_address
}
```

Sample vgwhttp.cfg File

Following is a sample `vgwhttp.cfg` file based on this information:

- You have a proxy server `caliber.myco.com` at port 9000
- The proxy server `caliber.myco.com` requires the credentials `admin1 psswd`
- You can directly access the web servers `quinn.myco.com` and `carly.myco.com`

Here is the `vgwhttp.cfg` file:

```
# vgwhttp.cfg - HTTP gateway configuration file
$control:1
proxy: caliber.myco.com 9000
{
    proxyAuth: admin1 psswd
    noproxy:
    {
        server: quinn.myco.com
        server: carly.myco.com
    }
}
$$
```

Indexing Network, UNC and Secure Paths on Windows

In order to index mapped network drives, UNC paths and secure file system paths with the File System gateway on Windows, you must ensure that you specified a user account and not the local system account for the Administration Server during installation and you must run `k2spider_srv.exe` as the same user with the rights described below.

For more information, see the *Verity K2 Installation and Setup Guide*.

Note When you index network mounted drives on Windows, you should use UNC paths (`\\host\path`) whenever possible as they provide a more reliable access method than mapped drive letters that may change after re-boots.

Running `k2spider_srv.exe`

You must run `k2spider_srv.exe` as a user with the following rights:

- act as part of the operating system
- increase quotas
- log on as a service
- replace process level tokens
- bypass traverse checking (this is set by default)
- create a token object

The user must be a local user, or a Domain user in a trusted Domain with local Administrator group privileges, and have the necessary access rights to the paths to be indexed.

Indexing Network, UNC and Secure Paths on UNIX

In order to index mapped network drives, UNC paths and secure file system paths with the File System gateway on UNIX, you must follow these instructions so that your K2 environment is properly configured.

The user account with which you are logged in when you run any of the command-line tools related to indexing (such as `k2admin`, `k2spider_srv`, `vspider`) must have the necessary access rights to the content being indexed.

If users and groups are on a host which supports shadow or md5 secure passwords, you must set the sticky bit for and grant root ownership to the `vspget` command-line tool before you run `k2spider_srv` or `vspider`. The `vspget` command-line tool is located in:

```
platformDir/bin/vspget
```

where *platformDir* is the directory in which you installed K2 Services, and *platform* will vary depending on your operating system.

To set the sticky bit for `vspget`, execute the following command:

```
chmod +s /platformDir/bin/vspget
```

You must substitute the appropriate value for *platformDir*.

To grant root ownership to `vspget`, execute the following command:

```
chown root /platformDir/bin/vspget
```

You must substitute the appropriate value for *platformDir*.

Defining the Authenticating Repository for Secure File System Collections

For secure file system collections, you must have a repository, or host name, that authenticates users attempting to search for and view documents in the collections. This authenticating repository is defined in the configuration file, `verity.cfg`.

By default, the repository name used to authenticate is the local host name. If you replicate a secure file system collection to other hosts, you should specify an independent authenticating repository.

To specify an independent authenticating repository, edit the `verity.cfg` file by uncommenting the `repositoryName:` line and adding the appropriate host name. For example:

```
#repositoryName needs to be specified if it needs to be machine
#or host independent
repositoryName: myname
```

Access Control List (ACL) Caching — Domain versus Local Users

By default, ACL caching is enabled for the File System gateway. This means file access information is cached and search users are authenticated against the cached information. When you secure files by local groups only in a Domain environment, ACL caching creates problems for authenticating Domain users.

To work around this problem, you can do one of two things:

- [Disable ACL caching for the File System gateway](#)
- [Secure Files with Domain Users and Groups](#)

Disable ACL caching for the File System gateway

Warning! Disabling ACL caching will result in slower performance during searches because the security check must be performed against the server for every file.

To disable ACL caching, follow these steps:

1. Make a copy of the `fssec` style set located in:

```
installDir/k2/common/styles/fssec
```

where *installDir* is the directory in which you installed K2 Spider

2. In the `vgwfsys.prm` style file in your copy of the `fssec` style set, add an octothorpe (#) as shown to the following line:

```
# $define VGWFSYS_SECURITY_ACLCACHE 1
```

3. Make sure the Domain users who need access to the collection have been added to a local group.

You are now ready to index the collection.

Secure Files with Domain Users and Groups

Use Domain users and groups to secure files, rather than local users or groups. This will allow Domain users to authenticate without problems.

Prefix Mapping

With prefix mapping, you can convert one field, such as a file system path, to another field, such as a URL. Using prefix mapping, you can index content by way of a file system, while viewing results content by way of a web server.

When mapping file system paths to URLs, the URLs must also be specified as aliases in your Web server.

K2 Spider Indexing Options

Option	Description
<code>prefixmap = file</code>	Specifies a control file which contains source field to destination field mapping information, See “Using the Control File” for more information on the content of this file.
<code>abspath = [true false]</code>	Forces the K2 Spider to use the absolute path to file system documents when indexing. This option is necessary to generate document paths that can be understood by the Web server, which would otherwise try to reconcile K2 Spider’s generated relative document paths using the Web server’s document root path as the starting point.

Using the Control File

The control file, which is specified by `prefixmap`, is a text file which contains the necessary information for mapping a source field to a destination field. A control file consists of the following columns:

Item	Description
<code>SourceField</code>	Field from which values for <code>SourcePrefix</code> will be read.
<code>SourcePrefix</code>	Specifies a prefix, such as a file system path, indexed by the K2 Spider. If the <code>SourcePrefix</code> is a path that includes a trailing slash, then <code>DestPrefix</code> must also include a trailing slash. If the path includes spaces, then enclose it in double quotes. For example: <code>SourcePrefix</code> <code>"C:\My Documents\Files"</code>
<code>DestField</code>	Field where the value modified by <code>DestPrefix</code> will be stored. Generally, you will be using the URL field for use with <code>\$\$doc.URL</code> in view templates.

Item	Description
DestPrefix	Specifies a destination prefix, such as a Web server alias, used by K2 Spider. If the DestPrefix is an alias, it must be created manually in the Web server, before users are allowed to view documents.
Flag	Maps backslashes to forward slashes when specified as a slash "/". This can be useful with indexed Windows file systems, where a SourcePrefix path ends with a trailing backslash and there are subdirectories.

Prefix Mapping Example 1

In this example, the control file `ctrlhtml.txt` maps a user's Web documents, indexed by way of file system, to a Web server alias for the user. It is assumed that `/search/html` is aliased to `/~search`, with `web` as the Web server document root, for the Web server. The relevant job INI entries would look like the following:

```
[Job]
start = /search/html
abspath = true
prefixmap = /verity/ctrlhtml.txt
```

Note that each record in the control file must be on a single line. The contents of `ctrlhtml.txt` would look like:

```
# For each document, all lines are considered in order.
#
# SourceField  SourcePrefix  DestField  DestPrefix
VdkVgwKey     /search/html  URL       http://web/~search
```

Prefix Mapping Example 2

In this example, the control file `ctrldoc.txt` maps Microsoft Word documents, indexed by way of file system, to a Web server alias. It is assumed that `d:\docs\isrel` is aliased to `/reldocs`, with `web/` as the Web server document root for the Web server. Furthermore, the Web server must be configured with a MIME Type for Microsoft Word documents. The command to run the K2 Spider would look like the following:

```
[Job]
start = d:\docs\isrel\
abspath = true
prefixmap = d:\verity\ctrldoc.txt
```

Note that each record in the control file must be on a single line. The contents of `ctrldoc.txt` would look like:

```
# For each document, all lines are considered in order.
#
#SourceField SourcePrefix DestField DestPrefix Flag
VdkVgwKey    d:\docs\isrel\  URL          http://web/reldocs/  /
```

Note that since `SourcePrefix` contains a trailing slash, `DestPrefix` does, also. As well, `Flag` is specified here because there may be subdirectories beneath `isrel` which need to have backslashes translated to forward slashes to be served properly by the Web server.

Prefix Mapping Example 3

In this example, PowerPoint slides have been converted to HTML. The conversion to HTML created two files for each slide; a text only `tsld*.htm`, and a text and graphics `sld*.htm`, where “*” is an incrementing number. We want to index the text only files, and then use the text and graphics files for links in a view template.

The control file, `ppview.txt`, will substitute a document file path, and then map the new path to a Web server alias. It is assumed that `c:\dev\slides` is aliased to `/~dev/slides`, with `web/` as the Web server document root, for the Web server. The command to run the K2 Spider would look like the following:

```
[Job]
start = c:\dev\slides
abspath = true
prefixmap = d:\verity\ppview.txt
```

Note that each record in the control file must be on a single line. The contents of `ppview.txt` would look like:

```
# For each document, all lines are considered in order.
#
#
SourceField  SourcePrefix  DestField  DestPrefix  Flag
VdkVgwKey   \dev\slides\t  URL        /dev/slides/  /
VdkVgwKey   c:\dev\slides\ URL        http://web/~dev/slides/ /
```

The first record in `ppview.txt` performs the path substitution required to have the text and graphics files, `sld*.htm`, used as links in a view template instead of the text only files, `tsld*.htm`. The rule of equal trailing slashes for `SourcePrefix` and `DestPrefix` can be disobeyed here because the `DestPrefix` is not a Web URL. The Web URL is mapped in the second record.

With `ppview.txt`, using `$$doc.URL` in a view template will correctly retrieve the text and graphics document from `c:\dev\slides\` by using the Web server alias `http://web/~dev/slides/`.

Setting MIME Types

You can use the MIME type criteria options `mimeinclude`, `indmimeinclude`, `mimeexclude` and `indmimeexclude` to include or exclude MIME Types.

- [Default MIME Types](#)
- [MIME Types Mapped by File Extension](#)
- [Complete List of Known MIME Types](#)
- [Indexing Unknown MIME Types](#)
- [MIME Types and Web Crawling](#)
- [MIME Types and File System Indexing](#)
- [Syntax Restrictions](#)

Default MIME Types

By default, there is a specific list of MIME Types recognized by K2 Spider. When indexing file systems, a file's extension determines the MIME Type. When indexing web sites, a file's extension can determine the MIME Type, but you can configure the relationship as well as add your own using the web server's administration utilities.

MIME Types Mapped by File Extension

When indexing file systems and web sites, the file extensions in the following table are automatically recognized by K2 Spider and are mapped to the corresponding MIME Types. Note that there may be several possible MIME Types for a particular document format. The table below lists the MIME Type mapped by K2 Spider.

For these extensions...	K2 Spider maps this MIME Type...
htm, html, shtml, asp, cgi, php, sml	text/html
txt, text, c, h, cpp, cxx, pl, eml	text/plain
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
pdf	application/pdf
mif	application/vnd.mif

For these extensions...	K2 Spider maps this MIME Type...
rtf	application/rtf
wpd	application/wordperfect5.1
aw	application/applixware
zip	application/zip
mbx	text/x-mbox

Complete List of Known MIME Types

Following is a list of all MIME Types known by K2 Spider by default. Where the previous table listed MIME Types known by file extensions, this is the complete list of MIME Types. Some correspond to the same file extension, while others do not have any known, default file extension.

- application/wita
- application/dec-dx
- application/dca-rtf
- application/x-mif
- application/vnd.mif
- application/rtf
- application/vnd.ms-works
- application/macwriteii
- application/wordperfect5.1
- application/vnd.ms-excel
- application/x-excel
- application/excel
- application/x-msexcel
- application/vnd.ms-powerpoint
- application/x-powerpoint
- application/power-point
- application/x-mspowerpoint
- application/msword
- application/x-ms-pst

- application/applixware
- application/pdf
- application/zip
- text/x-mbox
- text/html
- text/xml
- text/plain

Indexing Unknown MIME Types

Whenever you find MIME Types being dropped, or you know you will be indexing files whose extensions are not known to the K2 Spider by default, you can use the `mimemap` option or criteria to tune what is to be indexed.

Using mimemap

Use the `mimemap` option to refer to a file which contains your own custom mappings for file extensions and MIME Types.

The format for the control file used by `mimemap` is:

```
#file_ext_no_dot          mime-type
```

To map the unrecognized file extension `abc` to the MIME Type `application/msword`, you would do the following:

1. Create a text file, for example `mappings.txt`, that contains the following line.

```
abc                      application/msword
```

2. Include the `mimemap` option in your `job.ini` file, and refer to the text file that contains the appropriate information from step 1. For example:

```
mimemap = C:\mappings.txt
```

For more information, see [“Prefix Mapping” on page 352](#).

Using Criteria

You can use inclusion and exclusion criteria to finely control what is indexed.

- If your list of file types to index is long, use one of the exclusion criteria (`exclude`, `indexexclude`, `mimeexclude`, or `indmimeexclude`) to exclude extensions you know you do not want to index. For example:

```
exclude = '*.exe'  
exclude = '*.com'
```

- If the list of file types you want to index is relatively small, use one of the inclusion criteria (`include`, `indinclude`, `mimeinclude`, or `indmimeinclude`) to specify them. For example:

```
mimeinclude = 'text/*'  
include = '*.lst'  
include = '*.log'
```

MIME Types and Web Crawling

When you index a Web site, the K2 Spider evaluates your MIME type criteria against the "Content-Type" HTTP headers sent by the Web server hosting that Web site. That Web server passes along MIME type information based on its own internal tables.

When you encounter MIME types being dropped, make sure the Web server you are indexing has the necessary MIME type information. See the documentation for your Web server for information about specifying MIME types.

You can examine the indexing job's log files for indications that files are being skipped due to MIME types. For example, a typical ASCII file you might want indexed is a log file (`filename.log`). Unless the Web server understands that files with `.LOG` extensions are ASCII text, of MIME type `text/plain`, you will see in the indexing job log file that `.LOG` files are skipped because of MIME type even if you use:

```
mimeinclude = "text/*"
```

MIME Types and File System Indexing

When you index a file system, the K2 Spider reads file names and evaluates your MIME type criteria against an internal, compiled list of known MIME types and associated file extensions. You cannot edit this list. However, you can use the `mimemap` option to create a custom MIME type mapping.

When you encounter MIME types being dropped, check if the K2 Spider recognizes that particular MIME type. See the table, [“Complete List of Known MIME Types” on page 357](#).

You can examine the indexing job’s log files for indications that files are being skipped due to MIME types. For example, a typical ASCII file you might want indexed is a log file (`filename.log`). Since the K2 Spider does not understand that files with `.LOG` extensions are ASCII text, of MIME type `text/plain`, you will see in the indexing job log file that `.LOG` files are skipped because of MIME type even if you use:

```
mimemap = "text/*"
```

Syntax Restrictions

When you specify MIME type criteria, keep in mind the following restrictions.

Using the Wildcard Character (*)

The asterisk (*) wildcard character does not operate as a regular expression for the value of the MIME type criteria. Instead you can only use it to replace the entire MIME type or MIME sub-type.

For example, the following value is a valid substitute for `text/html`:

```
"text/*"
```

The following value is NOT a valid substitute for `text/html`:

```
"text/h*"
```

Multiple Parameter Values

You can use multiple instances of the MIME type criteria, each with a single parameter value. Quotes are necessary only if you use the wildcard character (*).

For example:

```
mimemap = text/plain  
mimemap = 'application/*'
```


Using StyleSet Editor in Standalone Mode

You can use the StyleSet Editor in standalone mode, independent of the K2 Dashboard, to create and update your style sets for the Lotus Notes Gateway, Documentum Gateway and Exchange Gateway.

For more information on using the StyleSet Editor in standalone mode, see the respective gateway guide:

- *Verity K2 Lotus Notes Gateway Guide*
- *Verity K2 Documentum Gateway Guide*
- *Verity K2 Exchange Gateway Guide*

Glossary

BIF	See bulk insert file .
browse	A command-line tool that lists the contents (field names and values) of a collection's document table.
bulk insert file (BIF)	A text file used to compile data into Verity internal formats. For example, a BIF is used to submit documents to a collection for indexing , or to insert queries into a Profile Net. Typically, BIFs are also used to insert or edit metadata not found in the source documents being indexed.
character map	A mapping between characters encoded in one character set and the same characters encoded in a second character set. Often, different character sets encode the same characters, because different standards were set up by different groups of people to encode the same set of characters from one character set to another without loss.
character set	A numeric encoding of the characters of a language. Text in a given language can be stored and manipulated using one or more character sets. The mapping occurs between characters and byte strings; that is, the combination of a particular character encoding (which maps between byte strings and integers) and a particular coded character set (which maps between integers and characters). Examples include ASCII, Shift-JIS, and UTF-8.

charset	In general terms, this is an abbreviation of character set . In K2, this term corresponds to the <code>-charset</code> argument, which sets a character set using a given command-line tool.
collection	The set of files and folders that stores all the information needed by VDK to search and classify documents in a repository . A collection stores the locations of all the indexed documents, the locations of all the indexed words in those documents, and metadata about the documents. It does not store the documents themselves.
collection indexing job	Specifies details for the indexing process, such as which documents to include in a collection , or the times when indexing should occur. These are also called K2 Spider jobs.
collection purge	The most common method for purging indexed documents. A collection purge removes all indexed content from the collection, leaving only the style files that contain the collection schema. This type of purge is necessary when style file changes are made that affect the schema. A collection purge automatically resets all indexing jobs that route information to only the specified collection. However, an indexing job can be set to route information to more than one collection, as in the case of mirrored collections. When a job routes information to more than one collection, you need to perform a job purge to properly reset it for indexing. See also job purge.
collection schema	The fields internal to a collection (as specified in <code>style.ddd</code> and any optional <code>style.ufl</code> files) and fields external to the collection (as specified by the gateway).
collection-level security	A security system that controls access to collections based on the user's identity. See also document-level security .
Controller	A K2 Spider Server process executed with the <code>-controller</code> option. A Controller manages Crawlers and Indexers and the gathering and distribution of indexing job information. Verity does not recommend adding <code>-crawler</code> or <code>-indexer</code> when you use <code>-controller</code>.

Crawler	A K2 Spider Server process executed with the <code>-crawler</code> option. A Crawler gathers document data for indexing jobs. A K2 Spider Server process can be executed with a combination of and multiple instances of the <code>-crawler</code> and <code>-indexer</code> options.
crawling	The process of seeking out documents in a repository to determine if they are valid candidates for indexing. If those documents contain links to other documents, K2 can be configured to follow those links and also crawl the target documents. Crawling is different from indexing. For example, K2 may be configured to crawl all HTML documents in a repository, and to follow any links to other documents, but to only index PDF documents it encounters.
didump	A command-line tool that generates a list of the words (tokens) in a collection's word index.
distributed search group	A search group in which some servers or brokers in the group are from a different domain.
document key	A unique identifier for each document in a collection. It identifies the gateway key and collection alias of a document.
document-level security	A method of controlling which documents appear in search results for a particular user. See also collection-level security.
end user	A person who uses K2 applications to search, browse, and retrieve information.
field	A discrete data item associated with the document such as the author, title, document location, creation date, etc.
gateway	A Verity software module used to retrieve documents from a specific type of repository for both indexing and viewing. K2 includes gateways for local files system, HTTP, Documentum, ODBC databases, MAPI (MS Exchange), and Lotus Notes.

gateway key	A unique identifier for any document or record in a collection . It is used to locate and retrieve the contents of the document. It is also known as the <code>VdkVgwKey</code> , short for Verity Development Kit Verity Gateway Key.
index	A Verity collection , knowledge tree , or parametric index .
Indexer	A K2 Spider Server process executed with the <code>-indexer</code> option. An Indexer inserts document data into a Verity collection. A K2 Spider Server process can be executed with a combination of and multiple instances of the <code>-indexer</code> and <code>-crawler</code> options.
index file	A file used by a search engine to locate specific web pages in a web site. The structure of an index file is similar in concept to the index of a book, where keywords are cross-referenced to their occurrence on pages.
indexing	A process that scans each document to be indexed and enters document text words and locations, as well as the metadata (title, author, size, internal zones, and so on) into a collection . See also crawling .
intellectual capital management	The process of combining human knowledge and experience (both implicit and explicit) with the information and data in an enterprise for the purpose of exploiting greater value.
job	See collection indexing job .
job purge	Purges the job that originally indexed the documents that now need removal from the collection. A job purge resets the particular job by removing its document records. It also removes from the associated collection(s) only those documents inserted by the job. A job purge is useful when the properties of one job are changed, and multiple jobs are associated with the collection. See also collection purge .
K2 Domain	A K2 system consisting of one Master Administration Server and all the K2 services (K2 Ticket Servers, K2 Brokers, K2 Servers, Administration Servers, and so on.) configured by that Master Administration Server. See also K2 system .

K2 Spider	A tool to perform spidering . K2 Spider performs crawling and, by default, also performs indexing . However, this indexing can be disabled and the crawled material can be indexed using another Verity tool. See also Verity Spider (vspider) .
K2 Spider Client	The executable and command-line tool used to interface with K2 Spider Servers to create and manage indexing jobs.
K2 Spider job	See collection indexing job .
K2 Spider Server	The executable and command-line tool that is run as either a Controller or some combination of Crawlers and Indexers.
K2 system	A generic term meaning a K2 installation. It may be either a K2 Domain or a K2 search group .
locale	A geographic or political region that shares the same language and customs.
locale ID (LCID)	A 32-bit value defined by Windows that consists of a language ID, a sort ID, and reserved bits.
metadata	Data that describes other data. For example, Author and Size could be metadata for a Microsoft Word document. Most metadata can be used in a Verity Query Language expression to search for documents containing the given metadata value.
mkprf	A command-line tool for building and maintaining Profile Nets.
mksyd	A command-line tool used to build a thesaurus from a thesaurus control file.
mktopics	A command-line tool for building and updating topic sets.
mkvdk	An all-purpose command-line collection maintenance tool.
normalization	An indexing feature in which a single version of a character is used when alternate versions exist (such as half-width and full-width kana in Japanese), and a single spelling is used for a word that has alternate spellings (such as <i>color</i> and <i>colour</i> in English). Users searching a normalized collection for a word find all words with either the common spelling or any of the alternate spellings.

ODBC	See Open Database Connectivity .
Open Database Connectivity (ODBC)	A middleware software standard that allows an application to communicate with various database engines.
operators	Reserved words that describe the relationship between search terms in the Verity Query Language .
partition	A subdivision of a collection . Partitioning collections improves scalability and searching performance.
primary document key	See document key .
purging	The act of deleting all records from a collection .
rcadmin	A command-line tool used to administer K2. It has similar functionality to the K2 Dashboard .
rck2	A command-line tool used to connect to K2 Servers for searching collections and other Verity indexes.
rcvdk	A command-line tool used for searching collections and displaying documents.
repository	A group of documents that are all stored in the same location, such as a file system. Other repositories can be organized in a relational database or a proprietary storage system such as Microsoft Exchange folders or Lotus Notes databases.
routing	The ability to specify which documents get indexed into which collections, based on characteristics (MIME type, document keys, or K2 Profiler matches) of the documents.
score	A numerical value indicating the degree of match between a document and a query. Scores, usually expressed to the end user as a decimal number between 0 and 1, are calculated during Verity search or Profiler operations. Scores are based on numerous factors, including the number of times search/query words appear in the document, their location in the document, and their proximity.

search application	The K2 components enabling administrators to index content and end-users to search and view that content. A search application typically includes K2 services (K2 Broker , K2 Server , K2 Ticket Server , and K2 Spider), JSP or ASP files, and Verity collections.
search engine	A software application that queries an index file . In broader terms it refers to a collection of programs which are used to index a repository and then present a user interface that enables queries to be constructed and searched.
spidering	The process of crawling and indexing . See also K2 Spider and Verity Spider (vspider) .
style file	A file used to configure the series of indexes in a collection that store data about its documents. The choice of a particular style file determines how indexing utilities function.
style.dft	A collection style file that controls the contents of the virtual document created during indexing.
style.fxs	A collection style file that contains feature-extraction stop words, that is, words that should not appear in document summaries and clusters.
style.lex	A collection style file that can control how tokenization occurs during indexing . Use of <code>style.lex</code> is discouraged; tokenization control is now available through the locale definition file associated with each locale.
style.prm	A collection style file containing parameters that control the generation of specialized indexes.
style.stp	A collection style file that contains indexing stop words, that is, words that should not be included in the collection's word index.
style.ufl	A collection style file that defines custom fields to be included in the collection's document table and optionally specifies the generation of indexes for those fields.
style.uni	A collection style file that controls the functioning of the universal filter .
style.zon	A filter style file that controls functioning of the zone filter .

Glossary

style set	The complete set of style files.
StyleSet Editor	The Verity application that enables administrators to create and modify style files.
VDK	See Verity Developer's Kit .
VdkVgwKey	See gateway key .
Verity Spider (vspider)	A command-line tool that provides indexing capabilities for many different document formats throughout the enterprise, including web-based, disk file systems, Lotus Notes, Microsoft Exchange, and ODBC. See also K2 Spider .

Index

Symbols

?

support for in URLs 287

850

setting with -charmap 194

8859

setting with -charmap 194

A

abspath option, Job section 254

-abspath, vspider 51

ACL caching

disabling 146, 351

domain versus local users 146, 351

Action option, FormsAuth section 268

agentname option, Network section 272

-agentname, vspider 61

aliases for file paths

-prefixmap 147

prefixmap 352

anchor text

gathering to improve relevancy 256

-anonymous

Crawler 185

Indexer 193

vsdb, vspider 94

vspider 67

any_field option, FormsAuth section 269

attachments

routing, by language 286

auth option

Job section 255

path change with getjobini 215

-auth, vspider 67

-auth2, vspider 68

authenticating

forms 268

authenticating repository

secure file system collections 145, 350

authorization

file for gateway, crawler 187

file for gateway, indexer 196

file for HTTP gateway 255

for Notes, Exchange, ODBC, Documentum
187

for Notes, Exchange, ODBC, Documentum
196

B

backslash, as literal character in BIF 230

bad count 238

badkey.log

unsupported MIME Types 90

badkeys.log

Controller 182

batch command, k2spider_cli 211

BIF

backslash, defining 230

character set 230

comments 228

control file parsing 229

create and submit 200

creating 228

document keys, unique 227

document-related fields, defining 227

end-of-document indicators 229

escape sequences 229

field definitions 228

not submitted 55

pathname interpretation 229

processing in-line 264

submitted 55

- submitting 219
- syntax example 231
- binding Crawlers and Indexers 280
- BoundCrawlIndx
 - DocRouter section 280
 - router type 280
- bulk insert file, see BIF
- bulk submit
 - using 169, 227
- by-passing proxy servers 278

C

- caching
 - of downloaded documents 60
- cand, VSDB status 245
- casesen
 - VSDB, k2spider_cli 243
 - vsdb, vspider 94
 - vspider 75
- casesen, Job section 256
- certfile
 - Crawler 185
 - indexing HTTPS 68
 - required by -keyfile 68, 185
 - required with -certfile 188
 - vspider 68
 - with SSL client certificates 68, 185
- certificate authority
 - importing the certificate 71
- CGI
 - support for 287
- cgiok, Criterion section 287
- cgiok, vspider 69
- character set
 - BIF 230
- charmap
 - Indexer 194
 - vspider 87
- Charset option, FormsAuth section 269
- client, K2 Spider

- commands 246
- starting 210
- using 209
- cmdfile
 - Controller 178
 - Crawler 185
 - Indexer 194
 - vspider 47
- coll
 - vsdb, vspider 94
- collectanchortext, Job section 256
- collection
 - Indexer 194
 - vsdb, vspider 94
 - vspider 48
- collections
 - creating 212
 - no optimizing 92
 - optimizing 119
 - purging keys 235
 - purging, vspider -purge 92
 - repairing, vspider -repair 93
 - style files 49
- collectlinkinfo, Job section 257
- combining processes
 - sharing -port option 181, 190, 202
- command file 178, 185, 194
- commands
 - grouping in a file 178, 185, 194
- comments, BIF 228
- common
 - Indexer 195
 - vsdb, vspider 95
 - vspider 87
- common directory
 - required with -license 179, 188, 197
- compact
 - vsdb, vspider 95
- configurations
 - considerations 175

Index

- crawlers with indexers 169
 - fully distributed 173
 - multiple processes 170
 - separate controller 171
- connect command, k2spider_cli 211
- connect to a Controller 211
- connection speed
 - all servers 175
 - to data repositories 175
- connections
 - Crawler 186
 - vspider 61
- considerations
 - data repositories 176
 - network connections 175
 - server resources 175
- Content-Length header, mapping 261
- control file
 - for -prefixmap 147
 - for prefixmap 352
 - parsing escape sequences 229
 - parsing in BIF 229
 - pathname interpretation 229
- Controller
 - logging bad keys 182
 - logging job errors 183
 - logging job status 182
 - logging skipped keys 182
- Controller, options
 - cmdfile 178
 - controller 179
 - help 179
 - ip 179
 - license 179
 - loglevel 180
 - msgdb 180
 - output 181
 - port 181
 - recover 181
 - workpath 182
- Controller, using 178
 - description of 162
 - getting status 236
 - sample output for getinfo 237
 - shutting down 219
- conventions
 - job.ini file 250
- convert
 - vsdb, vspider 95
- cookiefile, Job section 257
- cookies, Netscape
 - cookiefile 257
- crawled count 238
- Crawler
 - logging job errors 192
 - logging job status 192
- crawler
 - Crawler 186
- Crawler, in job.ini 270
 - crawlerspec option 270
- Crawler, options
 - anonymous 185
 - certfile 185
 - cmdfile 185
 - connections 186
 - crawler 186
 - gateway 186
 - gwauth 187
 - help 187
 - https 187
 - ip 187
 - keyfile 188
 - license 188
 - loglevel 188
 - msgdb 189
 - output 189, 201
 - port option 190
 - style 191
 - temp 191
 - workpath 192

- Crawler, using
 - and Indexers for file systems 176
 - binding with a Indexer 280
 - description of 162
 - getting status 236
 - maximum per Controller 270, 271
 - sample output for getinfo 237
 - sharing process space 184
 - shutting down 219
 - with Exchange 176
 - with Indexers for web servers 176
 - with Notes 176
 - with ODBC 176
- crawlerspec option, Crawlers section 270
- create command, k2spider_cli 212
- createcoll command, k2spider_cli 212
- creating
 - bulk files (BIF) 200
- Criterion section
 - cgiok option 287
 - detectdupfile option 288
 - domain option 288
 - exclude option 289
 - followdup option 289
 - followsymlink option 290
 - host option 290
 - https option 290
 - ignoreacl option 291
 - in job .ini file 287
 - include option 291
 - indexclude option 292
 - indinclude option 293
 - indmimeexclude option 294
 - indmimeinclude option 295
 - indskip option 296
 - javascriptok option 297
 - jumps option 298
 - lastmodified option 298
 - maxdocsize option 299
 - mimeexclude option 299
 - mimeinclude option 300
 - mindocsize option 300
 - nodocrobo option 301
 - nodupdetect option 301
 - nofollow option 302
 - norobo option 303
 - pathlen option 304
 - query option 305
 - refreshtime option 305
 - skip option 306
 - unlimited option 306
- D**
 - data repositories, considerations 176
 - databases
 - indexing 85
 - date
 - vsdb, vspider 95
 - date formats
 - for Last-Modified Date 122
 - datefmt
 - Indexer 195
 - vspider 87
 - dateformat, vsdb, vspider 95
 - day option, Schedule section 275
 - debug level logging 260
 - del, VSDB status 245
 - delay option, Network section 272
 - delay, vspider 62
 - delcoll
 - command, k2spider_cli 235
 - VSDB, k2spider_cli 243
 - vsdb, vspider 95
 - delete
 - VSDB, k2spider_cli 234, 243
 - vsdb, vspider 95
 - delete
 - command, k2spider_cli 214
 - deleted count 238
 - deleting

- jobs, description of 214
- jobs, instructions for 225
- keys with VSDB 234
- records with VSDB 235
- DestField, for prefixmap 352
- DestPrefix, for prefixmap 353
- detectdupfile option, Criterion section 288
- detectdupfile, vspider 52
- detecting duplicates 288
- direct-access hosts 64
- directories, counted as records 241
- disabling
 - maxmerge, maxclean, vdbopt 201
 - optimizing 201
- disabling ACL caching 146, 351
- disk cache 60
- DocRouter section
 - in job .ini file 279
 - routertype option 280
 - rule option 284
 - rulefile option 286
- docs not found 238
- document routing 280, 284, 286
 - BoundCrawlIdx 280
 - KeyBased 281
 - MimeBased 282
 - ProfileBased 283
 - RoundRobin 280
- document.write
 - crawling JavaScript 297
- document-related fields, defining 227
- documents
 - disabling caching 262
 - skipping by HTML tag 296, 306
 - specifying maximum size 299
 - specifying maximum to index 260
 - specifying maximum to index per host 260
 - specifying minimum size 300
 - submitting 220

- documentum gateway type 169
- documentum, vspider 50
- domain option, Criterion section 288
- domain, local users
 - and ACL caching 146, 351
- domain, vspider 69
- done, VSDB status 245
- duplicate detection 288
- duplicate documents
 - purging with vsdb 98
 - removing with vsdb 101
- duplicates
 - configuration considerations 176
 - detecting 288
 - disabling detection 301
 - following 289
 - following, vspider 70
 - preferred viewing 57

E

- encrypt
 - command, k2spider_cli 214
- encrypted passwords
 - in auth file, vspider 68
- end-of-document indicators, BIF 229
- ErrorURL option, FormsAuth section 268
- escape sequences, bulk insert 229
- evaluating include and exclude criteria 104, 314
- examples
 - createcoll command 213
 - indexing tasks 113, 323
 - prefix mapping 149–??, 150– 151, ??– 355
 - using -abspath 147
 - using abspath 352
 - using -prefixmap 147
 - using prefixmap 352
 - vsdb, vspider 98
- exchange gateway type 169
- Exchange Servers

- considerations 176
- gwauth for authorization 187, 196
- exclude option, Criterion section 289
- exclude, vspider 75
- extending the system 172

F

- fail, VSDB status 245
- field definitions
 - bulk insert 228
- file path resolution
 - to aliases 147, 352
- file system gateway
 - using -refresh 44
- file systems
 - considerations 176
 - indexing unknown MIME Types 358
 - known MIME Types, indexing 356
 - non-default MIME Types 156
 - overriding Last-Modified Date 130, 336
 - unknown MIME Types 155
- filesystems gateway type 169
- finddup
 - vsdb, vspider 95
- Flag, for prefixmap 353
- followdup, Criterion section 289
- followdup, vspider 70
- followsymlink, Criterion section 290
- followsymlink, vspider 70
- force threshold
 - ProfileBased rule 285
- FormsAuth section
 - Action option 268
 - any_field option 269
 - Charset option 269
 - ErrorURL option 268
 - in job .ini file 268
 - LoginURL option 268
 - Method option 269
- forms-based authentication 268

- free action
 - VSDB, k2spider_cli 243
- frequency, Schedule section 276

G

- gateway
 - Crawler 186
 - Indexer 195
- gateway type
 - documentum 169
 - exchange 169
 - filesystems 169
 - for Crawler 186
 - for Indexer 195
 - gateway, for OEMs 206
 - jobs, for 258
 - notes 169
 - notes and -anonymous 185, 193
 - web 169
- gatewaytype option, Job section 258
- gathering
 - but not indexing 55
- GDK
 - gateway type 206
- getinfo
 - command, k2spider_cli 215
 - sample output for crwl 237
 - sample output for ctrl 237
 - sample output for indx 238
 - using 236
- getjobini command, k2spider_cli 215
- getstat
 - command, k2spider_cli 215
 - sample output 238
 - using 238
- getting
 - Controller status 237
 - Crawler status 237
 - Indexer status 238
 - job state 215

-gwauth
 Crawler 187
 Indexer 196

H

hash sign (#), see pound sign
 header, Network section 273
 -header, vspider 62
 headers, mapping 261
 -help
 Crawler 187
 Indexer 196
 vspider 48
 help
 command, k2spider_cli 216
 output for k2spider_cli 246
 help file
 for k2spider_srv 179, 187, 196
 -help option 179
 hidden files
 indexed 176
 host option, Criterion section 290
 -host, vspider 70
 -hostcache, vspider 62
 hostname, Proxy section 277
 hosts
 using direct access 64
 hour option, Schedule section 275
 HTML documents
 and Last-Modified Date 111, 321
 reparsing for indexing 266
 skipping 296, 306
 HTTP header
 changing 273
 mapping 261
 HTTP request
 agent name, changing 272
 -https
 Crawler 187
 https option, Criterion section 290

-https, vspider 71

I

ignoreacl option, Criterion section 291
 ignoring
 ROBOT META directives 72, 301
 robots.txt 73, 303
 import, certificate
 using -https 71
 in collection, keys for VSDB 242
 inactive jobs
 starting 218
 include option, Criterion section 291
 -include, vspider 76
 indexclude, Criterion section 292
 -indexclude, vspider 77
 indexed count 238
 Indexer
 logging job errors 205
 logging job status 205
 Indexer, in job.ini 271
 -indexer, Indexer 196
 Indexer, options
 -anonymous 193
 -charmap 194
 -cmdfile 194
 -collection 194
 -common 195
 -datefmt 195
 -gateway 195
 -gwauth 196
 -help 196
 -indexer 196
 -indxsessions 197
 -ip 196
 -license 197
 -locale 198
 -loglevel 198
 -maxindfilehandles 199
 -maxindmem 199

- msgdb 199
- noindex 200
- nooptimize 201
- numpages 201
- port option 202
- style 203
- submitsize 203
- temp 204
- vdkclean 204
- workpath 204
- Indexer, using
 - and Crawlers for file systems 176
 - binding with a Crawler 280
 - description of 162
 - getting status 236
 - sample output for getinfo 238
 - sharing process space 193
 - shutting down 219
 - with Crawlers for web servers 176
- indexers, vspider 52
- indexerspec, Indexers section of job.ini file 271
- indexing
 - include and exclude criteria 104, 314
- indexing HTTPS
 - keyfile 71
- indexing, K2 Spider
 - attempting 273
 - disable caching of documents 262
 - examples 323
 - excluding MIME Types 294, 299
 - file paths 352
 - gather without submitting 263
 - including MIME Types 295, 300
 - maximum number of documents 260
 - maximum number of documents per host 260
 - network, UNC and secure paths 349
 - network, UNC and secure paths on Windows 348
 - Notes, with -anonymous 185, 193
 - overriding Last-Modified Date 336
 - proxy-related 344
 - reparsing HTML documents 266
 - restricting by domain 288
 - restricting by host 290
 - skipping HTML documents 296, 306
 - specifying file handles 199
 - specifying memory 199
 - SSL-enabled web sites 187, 290
 - starting points, specifying 267
 - UNIX file systems, symbolic links 290
 - using auth 339
- indexing, k2spider_cli
 - jobs, creating 222
 - jobs, deleting 225
 - jobs, modifying 224
 - jobs, pausing 224
 - jobs, resuming 224
 - jobs, starting 223
 - jobs, stopping 223
- indexing, vspider
 - and disk cache 60
 - examples 113
 - excluding MIME Types 79, 83
 - file paths 147
 - file systems 44
 - gather without submitting 55
 - including MIME Types 80, 84
 - mapping MIME Types 54
 - maximum number of documents 53
 - non-default MIME Types 156
 - overriding Last-Modified Date 130
 - proxy-related 140
 - reparsing HTML documents 45
 - restarting 41
 - restricting by domain 69
 - restricting by host 70
 - skipping HTML documents 81, 86
 - specifying file handles 53
 - SSL-enabled web sites 71

- starting points, specifying 41
- status reporting 94
- UNIX file systems, symbolic links 70
- using -auth 134
- indinclude, Criterion section 293
- indinclude, vspider 78
- indmimeexclude, Criterion section 294
- indmimeexclude, vspider 79
- indmimeinclude, Criterion section 295
- indmimeinclude, vspider 80
- indskip, Criterion section 296
- indskip, vspider 81
- indxsessions, Indexer 197
- ins, VSDB status 245
- inserted count 238
- ip option 179, 187, 196

J

- Java, starting K2 Spider
 - output, stdout and stderr 181

JavaScript

- support for crawling 297

- javascriptok, Criterion section 297

Job section

- abspath option 254
- auth option 255
- casesen 256
- collectanchortext 256
- collectlinkinfo 257
- cookiefile option 257
- gatewaytype option 258
- in job .ini file 254
- loglevel option 259
- maxnumdocs option 260
- maxnumdocsperhost option 260
- metamap option 261
- mimemap option 262, 358
- nocache option 262
- nosubmit option 263
- prefixmap option 263

- processbif option 264
- regexp option 265
- reparse option 266
- repos_charmap option 266
- start option 267

job syntax

- vspider 36

job.ini file

- conventions 250
- location, Controller 182
- location, Crawler 192
- location, Indexer 204
- options summary 252
- where sample is installed 252

joberror.log

- Controller 183
- Crawler 192
- Indexer 205

-jobpath 99

- and -purge 48
- vsdb, vspider 95
- vspider 48

jobs

- create 212
- creating 222
- delay until run again 276
- deleting 214
- enabling regular expressions 265
- frequency, specifying 276
- getting state 215
- getting statistics 238
- migrating, k2spider_cli 210
- modifying 216
- pausing 216
- purging records 217
- resume from pause 218
- running manually 274
- scheduling 274
- starting 218
- starting inactive 218

- stopping 218
- jobstatus.log
 - Controller 182
 - Crawler 192
 - Indexer 205
- jumps option, Criterion section 298
- jumps, vspider 71

K

- K2 Spider Client, starting 210

- k2spider_cli
 - help output 246
 - migrating 210
 - reporting 236
 - starting 210
 - using getinfo 236
 - using getstat 238
 - VSDB 221
- k2spider_cli commands
 - batch 211
 - connect 211
 - create 212
 - createcoll 212
 - delete 214
 - encrypt 214
 - getinfo 215
 - getjobini 215
 - getstat 215
 - help 216
 - modify 216
 - pause 216
 - purge 217
 - quit 217
 - resume 218
 - shutdown 219
 - start 218
 - stop 218
 - submit bif 219
 - submit doc 220
 - submit keys 220

- summary of 246
- k2spider_srv
 - as a Controller 178
 - as a Crawler 184
 - as an Indexer 193
 - default port number 178, 181, 184, 190, 193, 202
 - getting status 215, 236
 - options, summary 206
 - shutting down 226
 - shutting down components 219
- key
 - VSDB, k2spider_cli 244
 - vsdb, vspider 95
- KeyBased
 - DocRouter section 281
 - router type 281
 - rule, rulecriteria 284
- keycount
 - VSDB, k2spider_cli 244
- keyencode
 - vspider 88
- keyfile
 - Crawler 188
 - vspider 71
 - with SSL client certificates 188
- keys
 - for in collection for VSDB 242
 - from database versus from collection 241
 - lifecycle in persistent store 245
 - purging 235
 - submitting 220

L

- language, vspider 88
- language-based routing
 - and attachments 286
 - ProfileBased 285
- Last-Modified Date 111, 321
 - adding custom 130, 336

- customizing 130, 336
- how it is used 111, 321
- omitted by web server 130, 336
- overriding 130, 336
- See also lastmodified option
- valid formats 122
- Last-Modified header, mapping 261
- lastmodified option, Criterion section 298
- license
 - Controller 179
 - Crawler 188
 - Indexer 197
 - vspider 52
- license file
 - common directory 179
- limitation with PDF files
 - indxsessions 197
- links
 - gathering to improve relevancy 257
- locale
 - Indexer 198
 - vsdb, vspider 96
 - vspider 88
- location.href
 - crawling JavaScript 297
- log files
 - location 182, 192, 204
 - specifying level 180, 188, 198
- logging
 - debug level 260
 - skip level 259
 - specifying level for jobs 259
 - summary level 259
 - trace level 260
 - verbose level 259
- logging bad keys 182
- logging job errors
 - Controller 183
 - Crawler 192
 - Indexer 205

- logging job status
 - Crawler 192
 - Indexer 205
- logging skipped keys 182
- logging status
 - status.log, Controller 182, 192, 204
- LoginURL option, FormsAuth section 268
- loglevel
 - vspider 90
- loglevel
 - max. 500MB 259
- loglevel option, Job section 259
- loglevel, Controller 180
- loglevel, Crawler 188
- loglevel, Indexer 198
- logs
 - badkeys.log, Controller 182
 - joberror.log, Controller 183
 - joberror.log, Crawler 192
 - joberror.log, Indexer 205
 - jobstatus.log, Controller 182
 - jobstatus.log, Crawler 192
 - jobstatus.log, Indexer 205
 - skipkeys.log, Controller 182

M

- manually running jobs 274
- mapping
 - HTTP headers 261
 - MIME Types 262, 358
 - paths to Web aliases 263
- match
 - VSDB, k2spider_cli 244
 - vsdb, vspider 96
- matching gateway types 170, 171, 174
- matching indexers with collections 170, 172
- maxclean
 - disabling 201
- maxdocsize
 - not applicable to ZIP, PST 299

- maxdocsize option, Criterion section 299
- maxdocsize, vspider 82
- maxfilehandles, vspider 53
- maxindmem, Indexer 199
- maxindmem, vspider 53
- maxmerge
 - disabling 201
- maxnumdoc, vspider 53
- maxnumdocs, Job section 260
- maxnumdocsperhost, Job section 260
- messages
 - do not truncate 91
 - skip, due to redirection 90
- messages, log vspider 90
- metafile, vspider 82
- metamap
 - path change with getjobini 215
- metamap option, Job section 261
- Method option, FormsAuth section 269
- migrate
 - k2spider_cli 210
- migratevsdb
 - k2spider_cli 210
- MIME Types
 - and file system indexing 156
 - and web crawling 156
 - excluding 299
 - excluding from indexing 294
 - file system indexing 356, 360
 - for file system indexing 153
 - including 300
 - including for indexing 295
 - indexing unknown 155
 - mapping 262, 358
 - mapping with vspider 54
 - PST 357
 - setting 152, 356
 - syntax 360
 - unknown, indexing 358
 - unsupported written to badkey.log 90

- web site indexing 359
- MimeBased
 - DocRouter section 282
 - including ZIP 284
 - router type 282
 - rule, rulecriteria 284
- mimeexclude option, Criterion section 299
- mimeexclude, vspider 83
- mimeinclude option, Criterion section 300
- mimeinclude, vspider 84
- mimemap
 - path change with getjobini 215
- mimemap option, Job section 262, 358
- mimemap, vspider 54, 155
- mindocsize option, Criterion section 300
- mindocsize, vspider 84
- minute option, Schedule section 275
- mkvdk
 - using with -nooptimize 119
- modify command, k2spider_cli 216
- modifying jobs 216, 224
- month option, Schedule section 275
- msgdb
 - Controller 180
 - Crawler 189
 - Indexer 199
 - vspider 89

N

- Netscape cookies
 - cookiefile 257
- network connections
 - considerations 175
- network paths
 - K2 Spider 348, 349
- Network section
 - agentname option 272
 - delay option 272
 - header option 273
 - in job .ini file 272

- retry option 273
 - timeout option 273
- new action
 - k2spider_cli VSDB 242
- next action
 - VSDB, k2spider_cli 242
- nocache option, Job section 262
- nodocrobo option, Criterion section 301
- nodocrobo, vspider 72
- nodupdetect option, Criterion section 301
- nodupdetect, vspider 54
- noflowctrl, vspider 63
- nofollow option, Criterion section 302
- nofollow, vspider 72
- noindex
 - Indexer 200
 - vspider 55
- none, VSDB status 245
- nooptimize
 - Indexer 201
 - vspider 92
- noproxy, Proxy section 278
- noproxy, vspider 64
- norobo option, Criterion section 303
- norobo, vspider 73
- nosubmit option, Job section 263
- nosubmit, vspider 55
- notes gateway type 169
- Notes Servers
 - anonymous 185, 193
 - considerations 176
 - gwauth for authorization 187, 196
- notrunc
 - vspider options 91
- NTLM
 - indexing and viewing 339
- numpages
 - Indexer 201
 - vspider 56

O

ODBC

- considerations 176
 - gwauth for authorization 187, 196
 - indexing, vspider -query 85
- optimizing
 - by Indexer 201
 - disabling 201
 - with mkvdk 92, 119
- options
 - help for k2spider_srv 179, 187, 196
 - ip for k2spider_srv 179, 187, 196
 - job.ini file, summary of 252
 - k2spider_srv 206
- output
 - Controller 181
 - Crawler 189, 201
- overriding Last-Modified Date
 - file systems 130

P

- parent
 - vsdb, vspider 96
- password option, Proxy section 278
- Password, in job.ini
 - encrypting 214
- passwords
 - encrypted in auth file 68
- pathlen option, Criterion section 304
- pathlen, vspider 73
- pathname interpretation, bulk insert 229
- paths change, getjobini 215
- pause command, k2spider_cli 216
- pause, resuming jobs 218
- PDF
 - indexing limitation 197
- performance
 - connections option 186
 - file handles per Indexer 199

- file handles per indexing thread 53
 - memory for vspider 53
 - memory per Indexer 199
 - noflowctrl, vspider 63
 - submitsize, vspider 59
 - timeout for jobs, specifying 273
 - persist
 - vspider 56
 - persistent store
 - and platform dependence 30
 - lifecycle of a key 245
 - not updated with -noindex 200
 - purging 217
 - purging records 235
 - restoring with vsdb 103
 - port
 - Controller 181
 - Crawler 190
 - Indexer 202
 - sharing 181, 190, 202
 - portnum option, Proxy section 277
 - pound sign (#), BIF file 228
 - preferred
 - vsdb, vspider 96
 - preferred, vspider 57
 - prefix mapping 147, 352
 - abspath 254
 - using the control file 147, 352
 - prefixmap
 - path change with getjobini 215
 - prefixmap option, Job section 263
 - prefixmap, vspider 58
 - print
 - vsdb, vspider 96
 - processbif option, Job section 264
 - processbif, vspider 58
 - ProfileBased
 - DocRouter section 283
 - router type 283
 - rule, rulecriteria 284
 - proxy
 - K2 Spider, indexing and viewing 344
 - vspider, indexing and viewing 140
 - Proxy section
 - hostname option 277
 - in job .ini file 277
 - noproxy option 278
 - password option 278
 - portnum option 277
 - username option 277
 - proxy servers
 - authenticating 65
 - by-passing 278
 - specifying 65
 - proxy, vspider 65
 - proxyauth, vspider 65
 - pseudo-collection, from createcoll 213
 - PST
 - maxdocsize not applicable 299
 - MIME Type 357
 - purge
 - and -jobpath 48
 - using -jobpath 92
 - vspider 92
 - purge
 - command, k2spider_cli 217
 - keys and records with VSDB 235
 - records from persistent store 217
- ## Q
- query
 - ProfileBased routing 283
 - query option, Criterion section 305
 - query, vspider 85
 - quit action
 - k2spider_cli VSDB 243
 - quit command, k2spider_cli 217

R

- records
 - as directories 241
 - as total for VSDB 242
 - purging 235
 - purging from persistent store 217
 - showing with VSDB 245
 - status in K2 Spider job persistent store 245
 - versus keys 242
- recover
 - Controller 181
- recover
 - unfinished jobs 181
- recreate
 - and -anonymous 94
 - and -common 95
 - and -dateformat 95
 - and -locale 96
 - and -usegw 97
 - vsdb, vspider 96
- redirected URLs
 - not recorded as skipped 90
- refresh, vspider 44
- refreshtime option, Criterion section 305
- refreshtime, vspider 45
- regexp
 - VSDB, k2spider_cli 244
 - vspider 59
- regexp option, Job section 265
- regular expressions
 - enabling for VSDB, k2spider_cli 244
 - enabling in Job section 265
 - regexp, vspider 59
- relative paths
 - default 51
- relevancy
 - improving with anchor text gathering 256
 - improving with link gathering 257
- repair
 - vspider 93
- reparse option, Job section 266
- reparse, vspider 45
- repeat option, Schedule section 276
- repeating jobs, delay between 276
- reporting 236
 - getinfo 215
 - getstat 215
 - k2spider_cli VSDB 221
 - Verity spider status 94
 - vsdb 94
- repos_charmap
 - vspider 89
- repos_charmap option, Job section 266
- repositoryName, in verity.cfg
 - specifying an authenticating repository 145, 350
- restart
 - status of reparsed URLs 97
- restart, vspider 46
- restarting
 - vspider indexing jobs 41
- restricting indexing
 - by domain 288
 - by host 290
- resume command, k2spider_cli 218
- retry option, Network section 273
- retry, vspider 66
- ROBOT META directives, ignoring 301
- robots.txt, ignoring 303
- root ownership
 - setting for vspget 349
- RoundRobin
 - DocRouter section 280
 - router type 280
- routertype option, DocRouter section 280
- rule option, DocRouter section 284
- rulefile option, DocRouter section 286
- runtime.lic
 - in common 179, 188, 197

S

- samples
 - getinfo crwl 237
 - getinfo ctrl 237
 - getinfo indx 238
 - job.ini file 252
 - output for getstat 238
 - vgwhhttp.cfg for -auth 139
 - vgwhhttp.cfg for auth 343
 - vgwhhttp.cfg for proxy info 143, 347
 - VSDB session 241
- schedmode option, Schedule section 274
- Schedule section
 - day option 275
 - frequency 276
 - hour option 275
 - in job .ini file 274
 - minute option 275
 - month option 275
 - repeat option 276
 - schedmode option 274
 - year option 274
- scheduling jobs 274
- secure file system indexing
 - setting root ownership for vspget 349
 - setting sticky bit for vspget 349
- secure paths
 - K2 Spider 348, 349
- secure web sites
 - and vgwhhttp.cfg 255
 - auth entry in job.ini file 255
 - configuring viewing 136, 339
 - indexing, K2 Spider 339
 - indexing, vspider 134
 - NTLM, Basic and Public 339
- server resources, considerations 175
- setting MIME Types 152, 356
 - asterisk (*), using 356
 - multiple parameter values 152, 356
 - using the asterisk (*) 152
- sharing process space 184, 193
- showrec
 - VSDB, k2spider_cli 245
- shutdown command, k2spider_cli 219
- size
 - setting document minimum 300
- skip dup count 238
- skip level logging 259
- skip messages
 - do not include redirection 90
- skip option, Criterion section 306
- skip, vspider 86
- skipkeys.log
 - Controller 182
- SourceField
 - for prefixmap 352
- SourcePrfix
 - for prefixmap 352
- specifying character map
 - with Indexer 194
- SSL client certificates
 - using with -certfile option 68, 185
 - using with -keyfile 188
- SSL-enabled web sites
 - https for Crawler 187
 - indexing 290
- standalone mode
 - StyleSet Editor 158, 361
- start command, k2spider_cli 218
- start option, Job section 267
- start, vspider 41
- starting
 - jobs 223
 - K2 Spider Client 210
 - k2spider_cli 210
- starting points
 - file URLs 41
- statistics
 - for jobs with getstat 238
- status

- of URLs for a restart 97
 - VSDB, k2spider_cli 245
 - vsdb, vspider 97
- status
 - of k2spider_srv components 215
 - of records in for K2 Spider 245
- status.log
 - Controller 182, 192, 204
- stderr
 - redirecting to file 181
- stdout
 - redirecting to file 181
- sticky bit
 - setting for vspget 349
- stop command, k2spider_cli 218
- stopping jobs 223
- storing relative paths 51
- style
 - Crawler 191
 - Indexer 203
 - vspider 49
- style files
 - default location 49
 - with a Crawler 191
 - with an Indexer 203
- style.ddd file 227
- StyleSet Editor
 - in standalone mode 158, 361
- submit bif command, k2spider_cli 219
- submit doc command, k2spider_cli 220
- submit keys command, k2spider_cli 220
- submitsize
 - Indexer 203
 - vspider 59
- submitting BIF 58, 169, 200, 227
- summary level logging 259
- summary, VSDB output 241
- symbolic links
 - indexing in UNIX 70, 290
- synchronization

- with -noindex 55
- syntax
 - Mime types 360
 - vspider job 36
- system files
 - indexed 176

T

- temp
 - Crawler 191
 - Indexer 204
 - vspider 60
- temp directory
 - for k2spider_srv 191, 204
 - location 182, 192, 204
- threshold
 - ProfileBased routing 283
- timeout option, Network section 273
- timeout, vspider 66
- total
 - records for VSDB 242
 - versus in collection 242
- trace level logging 260
- truncating
 - log files 91
- tuning
 - connections option 186
 - delay for jobs, specifying 272
 - file handles per Indexer 199
 - file handles per indexing thread 53
 - memory per Indexer 199
 - retry for jobs, specifying 273

U

- UNC paths
 - K2 Spider 348, 349
- unfinished jobs, recovering 181
- unlimited option, Criterion section 306
- unlimited, vspider 74

- upd, VSDB status 245
- used, VSDB status 245
- useget, vspider 74
- usegw
 - vsdb, vspider 97
- username option, Proxy section 277

V

- vdbopt
 - disabling 201
- vdkclean
 - Indexer 204
- VdkVgwKey
 - style.ddd file 227
- verbose level logging 259
- Verity spider reporting
 - vsdb 94
 - vsdb examples 98
- verity.cfg
 - repositoryName 145, 350
- vgwhttp.auth
 - decrypting password 68
- vgwhttp.cfg
 - proxy section 142, 346
 - repository section 137, 341
 - sample for -auth 139
 - sample for auth 343
 - sample for proxy info 143, 347
 - viewing secure web sites, K2 Spider 340
 - viewing secure web sites, vspider 136
 - viewing through proxy, K2 Spider 345
 - viewing through proxy, vspider 141
- vgwhttp.cfg file
 - enabling secure viewing 255
- viewing
 - preferred 57
 - through proxy, K2 Spider 345
 - through proxy, vspider 141
 - with auth, K2 Spider 340
 - with auth, vspider 136
- virtual hosts
 - indexing, vspider 74
- virtualhost, vspider 74
- VSDB
 - command, k2spider_cli 221
- vsdb
 - purging duplicate documents 98
 - removing duplicate documents 101
 - restoring a persistent store 103
- vsdb arguments, vspider
 - anonymous 94
 - casesen 94
 - coll 94
 - collection 94
 - common 95
 - compact 95
 - convert 95
 - date 95
 - dateformat 95
 - delcoll 95
 - delete 95
 - finddup 95
 - jobpath 95, 99
 - key 95
 - locale 96
 - match 96
 - parent 96
 - preferred 96
 - print 96
 - recreate 96
 - status 97
 - usegw 97
- VSDB, k2spider_cli
 - casesen 243
 - delcoll 243
 - delete 243
 - enabling regular expressions 244
 - free action 243
 - key 244
 - keycount 244

Index

- match 244
- new action 242
- next action 242
- quit action 243
- regexp 244
- sample session 241
- showrec 245
- status 245
- summaries 242
- summary differences 241
- total and in collection 242
- using 240
- using -delcoll 235
- using -delete option 234
- vsdb, vspider
 - examples 98
 - using 94
- vspget
 - setting root ownership 349
 - setting the sticky bit 349
- vspider options
 - abspath 51
 - agentname 61
 - anonymous 67
 - auth 67
 - auth2 68
 - casesen 75
 - certfile 68
 - cgiok 69
 - charmap 87
 - cmdfile 47
 - collection 48
 - common 87
 - connections 61
 - datefmt 87
 - delay 62
 - detectdupfile 52
 - documentum 50
 - domain 69
 - exclude 75
 - followdup 70
 - followsymlink 70
 - header 62
 - help 48
 - host 70
 - hostcache 62
 - https 71
 - include 76
 - indexexclude 77
 - indexers 52
 - indinclude 78
 - indmimeexclude 79
 - indmimeinclude 80
 - indskip 81
 - jobpath 48
 - jumps 71
 - keyencode 88
 - keyfile 71
 - language 88
 - license 52
 - locale 88
 - loglevel 90
 - maxdocsize 82
 - maxfilehandles 53
 - maxindmem 53
 - maxnumdoc 53
 - metafile 82
 - mimeexclude 83
 - mimeinclude 84
 - mimemap 54, 155
 - mindocsize 84
 - msgdb 89
 - nodocrobo 72
 - nodupdetect 54
 - noflowctrl 63
 - nofollow 72
 - noindex, vspider 55
 - nooptimize 92
 - noproxy 64
 - norobo 73

- nosubmit 55
- numpages 56
- pathlen 73
- persist 56
- preferred 57
- prefixmap 58
- processbif 58
- proxy 65
- proxyauth 65
- purge 92
- query 85
- refresh 44
- refreshtime 45
- regexp 59
- repair 93
- reparse 45
- repos_charmap 89
- restart 46
- retry 66
- skip 86
- start 41
- style 49
- submitsize 59
- temp 60
- timeout 66
- unlimited 74
- useget 74
- virtualhost 74
- vspider options, old
 - debug 91
 - trace 91
 - verbose 91
- vspider, using
 - and proxy access 64
 - indexing job syntax 36

W

- web gateway type 169
- web indexing
 - and MIME types 359

- web servers, network considerations 176
- window.open
 - crawling JavaScript 297
- Windows
 - hidden files, indexed 176
 - system files, indexed 176
- work path
 - set temp directory 191, 204
- workpath
 - Controller 182
 - Crawler 192
 - Indexer 204

Y

- year option, Schedule section 274

Z

ZIP

- maxdocsize not applicable 299
- must be declared with MimeBased 284