Verity K2 rcadmin Guide

Version 6.1

November 30, 2005
Part Number DE0270

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

Verity Benelux BV
Coltbaan 31
3439 NG Nieuwegein
The Netherlands

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 Unpublished-rights reserved under the copyright laws of the United States.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.

11/8/05

Contents

Preface 11
Using this Book 12
Version 12
Organization of this Book 12
Stylistic Conventions 13
Related Documentation 14
Verity Technical Support 14

1 Getting Started 17
Introducing rcadmin 18
Introducing Administration Servers 18
Administration XML Files 19
Before You Begin 20
Starting an Administration Server Manually 20
Starting an Administration Server Manually on UNIX 21
Starting a K2 Server Manually 21
Starting rcadmin 22
Logging in 23
Logging in Automatically 24
Obtaining Help 25
Setting Values 25
Understanding K2 Error Messages 25
Restarting the K2 Server 26
Exiting rcadmin 27

Automating Administration Tasks 27
Writing Commands to a Log File 27
Using Existing Scripts with the Current Version of rcadmin 28
Using the version Command 28
Revision History 30
Revisions in K2 Version 6.1 30
Revisions in K2 Version 6.0 30
New Commands 30
Modified Commands 31
Revisions in K2 Version 5.5 33
New Commands 33
Modified Commands 34
Deprecated Commands 36

2 K2 System Commands 37

Viewing Services and Indexes in the K2 System 38
hierarchyview 38
hostview 39
indexview 39
Managing Licenses 40
licenseset 41
licenseget 42
licensecheck 45
licenseservice 47
licenseupdate 47
servercpuset 48
Managing K2 Domain Names 49
hostdomainset 49
hostdomainget 50

3 Managing K2 Services 51

Adding a K2 Service 52
Configuring Service Logs 53
Using Trusted Clients to Control Connections 53

Managing K2 Administration Servers 54
adminset 54
adminget 56
Managing K2 Brokers 58
brokerset 59
brokerget 61
Managing K2 Servers 64
serverset 65
serverget 68
vdksettingset 73
Managing K2 Ticket Servers 75
ticketset 76
ticketget 79
Managing K2 Index Servers 82
Adding a K2 Index Server 82
Starting a K2 Index Server 85
Registering Indexes for Use with a K2 Index Server 85
indexerset 86
indexerget 88
Managing External K2 Services 90
extbrokerattachset 91
extticketattachset 94
extsvcticketattachset 97
extbrokerdetach 99
extticketdetach 101
extsvcticketdetach 102
Attaching and Detaching Services 104
servicedetach 105
brokerattachset 107
brokerattachlistget 109
brokerattachget 110
ticketattachset 111
ticketattachlistget 112
ticketattachget 113

Starting and Stopping K2 Services 115
adminlaunch 116
adminsignal 117
adminkilllaunched 118
servicesignal 119
servicestartup 120
servicewatchedstate 121
servicegetpid 121
wsadd 122
wsdel 123
wsdelall 123
wsget 123
Checking the State of a K2 Service 125
servicecheckstate 125
servicestateget 126
Removing K2 Services 127
servicedel 127
Managing Recommendation Transaction Logs 129
retxlogset 129
Configuring the Search Results Cache 131
searchcacheset 131

4 Managing Profile Services 133

Introduction 134
Configuring a Profile Service 135
profileset 135
profileget 138
profiledel 140
Configuring a Profile Net 141
profilenetset 141
profilenetget 143
profilenetdel 144
profilenetstateset 145

5 Managing Logging 147

Introduction 148

Managing Status Logs 148

statuslogset 149

statuslogget 151

Managing Query Logs 152

querylogset 152

querylogget 156

querylogdel 158

querylogexcludeset 159

querylogexcludedel 160

querylogexcludeget 160

6 Managing Security 163

Managing Trusted Clients 164

trustedclientset 165

trustedclientget 166

trustedclientdel 167

Managing Login Modules 168

Configuring the Login module 168

ticketlmset 170

ticketlmget 176

Managing Persistent Storage 181

ticketpsmset 182

ticketpsmget 185

7 Managing Indexes 189

Introduction 190

Registering Indexes 190

Basic Commands 191

indexdel 192

indexdetach 193

indexattach 194

indexserverget 195

indexstateset ... 197
indexvdkset ... 198
indexviewset ... 200
styleset ... 201
styleget ... 202
styledel ... 204
Managing Collections ... 205
Adding a Collection to a K2 System ... 205
collset ... 206
collcreate ... 209
collget ... 211
collstatget ... 214
collpurge ... 217
Managing Knowledge Trees ... 218
Adding a Knowledge Tree to a K2 System ... 218
treeset ... 219
treeget ... 221
Managing Topic Sets ... 224
topicsetset ... 225
topicsetget ... 226
topicsetcollattach ... 227
topicsetcolldetach ... 227
Managing Parametric Indexes ... 228
Adding a Parametric Index to a K2 System ... 228
paraset ... 229
paraget ... 231
parastatget ... 232
Managing Recommendation Indexes ... 234
Adding a Recommendation Index to a K2 System ... 235
Recommendation Index Types ... 236
reset ... 237
reget ... 240
riset ... 242
riget ... 244
risourceset ... 245

risourceget 248
risourcedel 249
ritypeset 250
ritypeget 252
ritypedel 253
rityperelset 254
rityperelget 256
rityperelde l 257

8 Managing Indexing Jobs 259

Managing K2 Spider 260
Adding a K2 Spider to a Host 260
spiderset 261
spiderget 262
spiderdel 263
Managing Indexing Jobs 264
Creating a Job Chain 265
jobcreate 267
jobmodify 271
jobgetdetail 273
jobgetstats 275
jobstart 278
jobstop 278
jobpause 279
jobresume 279
jobpurge 280
jobdel 281
jobaddindex 282
jobremoveindex 283

9 Managing User Defined Jobs 285

Introduction 286
Creating a User Defined Job 286
Custom Job Types 287
Creating a Job Chain 287

Managing User Defined Jobs 289
udjcreate 289
udjmodify 294
udjget 296
udjdel 298
Controlling User Defined Jobs 299
udjstart 299
udjstop 300
udjabort 300
Managing Custom Job Types 301
udjsset 301
udjsget 306

Index 311

Preface

This guide is for readers who want to administer their K2 system using a command-line tool, and may want to create scripts to maintain their system. The `rcadmin` tool provides a command-line alternative to the K2 Dashboard. The K2 Dashboard is Verity's primary interface for K2 administration, and is described in the *Verity K2 Dashboard Administrator Guide*. Before reading the *Verity K2 rcadmin Guide*, you should read the *Verity K2 Getting Started Guide*, and the overview sections of the *Verity K2 Dashboard Administrator Guide*. Those documents explain the concepts and architecture of the K2 system.

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 6.1. The content was last modified November 30, 2005. Corrections or updates to this information may be available through the Verity Customer Support site; see "Verity Technical Support" on page 14.

Organization of this Book

This book includes the following parts, chapters and appendixes:

- Chapter 1, "Getting Started" provides an overview of K2 administration and the `rcadmin` command-line tool.
- Chapter 2, "K2 System Commands" describes commands that manage the K2 system.
- Chapter 3, "Managing K2 Services" describes commands that manage K2 services.
- Chapter 4, "Managing Profile Services" describes commands that manage Profile Services.
- Chapter 5, "Managing Logging" describes commands that manage K2 message logs.
- Chapter 6, "Managing Security" describes commands that manage security.
- Chapter 7, "Managing Indexes" describes commands that manage K2 indexes.
- Chapter 8, "Managing Indexing Jobs" describes commands that manage K2 Spider indexing jobs.
- Chapter 9, "Managing User Defined Jobs" describes commands that manage user defined jobs.

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.
Italics	Book titles and new terms: ■ For more information, see the *Verity K2 Getting Started Guide*. ■ An *index* is a Verity collection, parametric index, or knowledge tree.
`Monospace`	File names, paths, code, and required user input: ■ The `name.ext` file is installed in: `C:\Verity\Data\`
`Monospace italic`	Replaceable strings in file paths and code: ■ `user` *`username`*
`Monospace bold`	Data types: ■ **`SrvConnect`** A connection handle.

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.

Convention	Usage
`required`	Absence of braces or brackets indicates required syntax in which there is no choice; you must enter the required syntax element.
`variable`	Italics specify variables to be replaced by actual values, as in `-merge` *`filename1`*
`...`	Ellipses indicate repetition of the same pattern, as in `-merge` *`filename1`*`,` *`filename2`* `[,` *`filename3`* `... ]` where the ellipses specify *`filename4`*, 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 the K2 architecture and K2 Administration:

- *Verity K2 Getting Started Guide*
- *Verity K2 Dashboard Administrator 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 at 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 need to supply your Verity entity ID and Verity license key.

1

Getting Started

This chapter introduces K2 administration and `rcadmin`, a command-line tool that you can use to configure your K2 system.

- Introducing rcadmin
- Before You Begin
- Starting rcadmin
- Logging in
- Obtaining Help
- Setting Values
- Understanding K2 Error Messages
- Restarting the K2 Server
- Exiting rcadmin
- Automating Administration Tasks
- Revision History

Introducing rcadmin

`rcadmin` enables you to view and update configuration settings for the components (K2 Servers, K2 Brokers, collections, and so on) in your K2 system. It provides a command-line alternative to the K2 Dashboard browser-based user interface. In some cases, it can also configure settings that cannot be configured through the K2 Dashboard.

`rcadmin.exe` is stored in the directory *`platformDir`*`/bin`, where *`platformDir`* is the pathname of the directory containing platform-specific executable code (for example, `C:\Program Files\Verity\k2\nti40` on Windows).

Introducing Administration Servers

A K2 *Administration Server* is a repository for configuration information. In a K2 domain, there is one Master Administration Server or Administration Server for every host. An Administration Server services the administration needs on a host and synchronizes with the Master Administration Server. K2 Administration Servers fill three important roles:

- They act as repositories of configuration information for your K2 system.
- They transfer files used by the K2 system from one host to another.
- They monitor K2 components, such as K2 Servers, K2 Brokers, and K2 Ticket Servers, and start/stop them as necessary. This includes automatically re-launching them if they fail.

K2 uses one Master Administration Server and possibly one or more Administration Servers. The *Master Administration Server* is the central hub for configuration information. A K2 domain must have one and only one Master Administration Server.

When you install or remove K2 components, or configure the K2 system, K2 updates the configuration information in the Master Administration Server. The Master Administration Server then echoes it to all appropriate Administration Servers in the system. This ensures that all Administration Servers know their current state and their relationship to the K2 system.

All administration changes should be made by connecting to the Master Administration Server, and specifying the alias of the Administration Server you want to modify.

Note If the Master Administration Server is not available, you can make changes on a particular Administration Server, but those changes will be overwritten when the Master Administration Server reconnects and synchronizes its configuration with the Administration Server.

For more information on Administration Servers, see the *Verity K2 Dashboard Administrator Guide*.

Administration XML Files

All Administration Servers (including the master) store their configuration information in XML files named `admin`*`N`*`.xml`, where *`N`* is a number from 0 to 9. Each time a configuration change is made, K2 writes out a new file and uses the next number in the sequence, starting over at 0 after 9 is used. This means that the latest file may have any number from 0 to 9. You cannot assume that `admin9.xml` is always the current file.

Note Use the `adminget` command to determine the XML file that is currently used. See "adminget" on page 56.

To back up the administration XML file use the `label` command. At the `rcadmin` prompt, type `label`, and then enter the path to the backup file. For example, on Windows it might be: `C:\temp\backup.xml`. For information on restoring this backup file, see the *Verity K2 Dashboard Administrator Guide*.

WARNING! Do not edit or alter the XML files unless instructed to do so by Verity Technical Support. Doing so can render your K2 system unusable. To configure your system, use only the K2 Dashboard or `rcadmin`.

These files are stored in the directory *`dataDir`*`/host/admin`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). They are written to disk within about ten seconds after making a change via `rcadmin`. The Master Administration Server then echoes the changed file to all the appropriate Administration Servers. This ensures that they all have the most current information in case the Master Administration Server is unavailable or the connection breaks.

For more information about `admin`*`N`*`.xml` files, see the *Verity K2 Dashboard Administrator Guide*.

Before You Begin

Before you start `rcadmin`, you must:

- Ensure the K2 services components are installed. This includes a Master Administration Server and `rcadmin`. For more information, see the *Verity K2 Installation and Setup Guide*.
- Ensure the Master Administration Server is running.
- Ensure the K2 Ticket Server is running, if at installation you configured the K2 Dashboard to run with a K2 Ticket Server.

Starting an Administration Server Manually

To start an Administration Server, type the following at the command prompt:

`k2admin [option1 option2 ...]`

The available options are:

Keyword	Permitted Values	Function
`-cfg filename`	Any valid path and filename. For example, `C:\Special\new.cfg`.	The configuration file to use. By default, this is `verity.cfg`.
`-xmlFile filename`	Any valid path and filename	The `adminN.xml` file to use.
`-dtdFile filename`	Any valid path and filename	The DTD file associated with the `adminN.xml` file.
`-autorollback value`	`0` = disable `1` = enable	Enables or disables the auto-rollback function. See the *Verity K2 Dashboard Administrator Guide* for more information.

Keyword	Permitted Values	Function
`-VCM` *`value`*	`0` = disable `1` = enable	Enables or disables the Verity Control Module (VCM), which allows K2 to stop or restart watched services as necessary.
`-ntService` *`value`*	`0` = remove `1` = install	If set to 1, the Administration Server runs as a service on Windows. Otherwise, it runs as a regular application.
`-servicename` *`name`*	Any name	The Windows service name. This only has an effect if `-ntService` is set to 1. If this is not specified, the default name is "Verity K2 Administration Server."

Starting an Administration Server Manually on UNIX

To start the Administration Server on UNIX, run the script `k2adminstart.sh`.

To stop the Administration Server on UNIX, run the script `k2adminstop.sh`.

The scripts are stored in the directory *`platformDir`*`/bin`, where *`platformDir`* is the pathname of the directory containing platform-specific executable code (for example, `C:\Program Files\Verity\k2\nti40` on Windows).

Starting a K2 Server Manually

A K2 Server is started by using the following command:

`k2server [`*`option1 option2 ...`*`]`

The available options are:

Keyword	Permitted Values	Function
`-alias` *`name`*	Any name.	The alias of the K2 Server.
`-charmap` *`name`*	The name of any valid character map.	The character map that the K2 Server should use.

Starting rcadmin

To start `rcadmin`, open a DOS or shell window:

- To connect to the Master Administration Server, type:

  ```
  rcadmin
  ```

- To connect to an Administration Server using the server's alias, type:

  ```
  rcadmin -alias adminAlias
  ```

  where *`adminAlias`* is the Administration Server's alias.

- To connect to the Administration Server on the local machine (the machine where `rcadmin` is running), type:

  ```
  rcadmin @local
  ```

- To connect to the Administration Server using the server name and port number, type:

  ```
  rcadmin -server server_name -port port_number
  ```

  This is useful when you do not remember the alias, or when the Master Administration Server is unavailable.

- To start `rcadmin` and specify the character set `rcadmin` will use for configuration information, type:

  ```
  rcadmin -charmap character_set
  ```

  For example, to specify the character map Shift-JIS, type:

  ```
  rcadmin -charmap sjis
  ```

  See the *Verity Locale Configuration Guide* for more information on character sets.

After starting `rcadmin`, you will see the `rcadmin` command prompt:

```
K2Admin Server: adminserver1:9950 connected. Type ? for help.
RCADMIN - Verity, Inc. Version 6.1
rcadmin>
```

Logging in

If you are using a K2 system with a K2 Ticket Server, you must log in before entering a command. Most `rcadmin` commands require *authentication* from the K2 Ticket Server. Authentication is the process of passing credentials to a secure server, such as a K2 Server, K2 Broker, or K2 Ticket Server. If you have not logged in, you may receive a `-1207` error when you try to save changes to the system.

The `getgroup`, `errormessage`, `help` and `?` commands work *without* logging in.

Note If authentication is enabled on an external domain, you must log in to the external domain to update any external services. See "Managing External K2 Services" on page 90 for more information.

To log in to `rcadmin`:

1. Ensure you are defined as an *administrator*. See the *Verity K2 Dashboard Administrator Guide* for more information.
2. Type `login` or `login` *`username`* at the `rcadmin` prompt.
3. Enter your K2 administrator user name at the `User` prompt.
4. Enter your K2 administrator password at the `Password` prompt.
5. Optionally, enter a Windows domain name at the `Windows NT Domain` prompt.

To log out of `rcadmin`, type `logout` at the `rcadmin` command prompt.

To view a list of the groups to which the currently logged in `rcadmin` user belongs, type `getgroup` at the `rcadmin` command prompt.

To check whether a user who is currently logged in to `rcadmin` is a valid user, type `checkuser` at the `rcadmin` command prompt.

Logging in Automatically

You can also log in automatically by saving login credentials (host name, the machine user name, and the K2 Administration user name and password) in an `admincred.dat` file. This file is used when you want to script `rcadmin`, but do not want to expose passwords in the scripts, or when you *swap* collections using K2 Dashboard. If you frequently need to update collections, and you want to take collections offline for as little time as possible, you can use the K2 Dashboard to "swap" collections. See the *Verity K2 Dashboard Administrator Guide* for more information on collection swapping.

When you start `rcadmin`, the program looks for the `admincred.dat` file in the directory *`installDir`*`/k2_6/k2/common`, where *`installDir`* is the pathname of the Verity installation directory (for example, `usr/verity/` on UNIX, or `C:\Program Files\Verity` on Windows). If the file exists, `rcadmin` verifies that the host name and the machine user name in the file matches the host name and user name of the machine on which the file is stored. This prevents the file from being used on a machine other than the one on which it was created. It then automatically tries to log in using those credentials. If `rcadmin` cannot log in using the credentials or cannot find the file, `rcadmin` starts without logging in. You must then log in manually using the `login` command.

To create an `admincred.dat` file, follow these steps:

1. Start `rcadmin`.
2. At the `rcadmin` prompt, type `login -save`.
3. Enter your K2 administrator user name at the `User` prompt.
4. Enter your K2 administrator password at the `Password` prompt.
5. Optionally, enter the Windows domain name at the `Windows NT Domain` prompt.

An `admincred.dat` file is created in the directory *`installDir`*`/k2_6/k2/common`. The next time you start `rcadmin`, you are logged in automatically.

Note If an `admincred.dat` file already exists, it will be overwritten with the new login information.

Note Do not change the name or the location of the file. It must be named `admincred.dat`, and must be stored in the directory *`installDir`*`/k2_6/k2/common`.

Obtaining Help

To view a list of `rcadmin` commands, type one of the following at the `rcadmin` prompt:

`?` Shows a summary of all commands.

`help` Shows a menu of administration tasks. Select the number of the task for which you need help. The commands in the selected task group are shown.

Setting Values

When you enter values at an `rcadmin` prompt, `rcadmin` often shows the current settings in parentheses after the prompt. To accept this setting, press the `Enter` key.

To remove a string setting, such as a server alias or index alias, press the space bar, and press the `Enter` key.

After entering most commands, `rcadmin` prompts you for additional information. If you are modifying the K2 configuration, you are prompted to save the changes:

```
Save changes? [y|n]:
```

Type `y` to save your changes, or `n` to cancel.

Understanding K2 Error Messages

If you attempt an operation in a K2 command-line tool, and it fails, note the code number reported in the error message and use the `errormessage` command in `rcadmin` to translate the K2 error code into a text description of the error.

1. Log into `rcadmin`.
2. At the `rcadmin` command prompt, enter the `errormessage` command.
3. At the `Error Number` prompt, enter the K2 error number. For example, `-1204`. `rcadmin` shows a description of the error number if one is available.

Note Some K2 errors may not have a corresponding description in the Verity message database.

For example:

```
C:\>rcadmin
K2Admin Server: host.verity.com:13300 connected. Type ? for help.
rcadmin> login
User:adminaccount
Password:********
Windows NT Domain (optional):
User <adminaccount> logged in.
rcadmin> errormessage
Error code: -12483
Description: TCP/IP Error: Call Connect.
rcadmin>
```

Restarting the K2 Server

When you modify some parameters, you must restart the K2 Server, using the `servicesignal` command, for the change to take effect. The modification may require a *quick* or *full* restart.

A quick restart resets the K2 Server with minimal impact to its service. During a quick restart, the K2 Server re-reads the settings in the local K2 Administration and updates its configuration. The K2 Server does not have to stop. Client connections are locked and clients may experience a brief service delay.

A full restart shuts down the K2 Server and then starts it back up so that a changed parameter can take effect. You will experience a brief outage while the K2 Server shuts down and restarts.

If a command or parameter requires a restart, the type of restart required is noted in the field description.

Exiting rcadmin

To exit `rcadmin`, type any of the following:

```
q
quit
exit
```

Automating Administration Tasks

You can create scripts using a scripting language and `rcadmin` commands to automate many administration tasks. For example, you can create a script to start or stop a K2 Server, add a new collection, or take a index offline.

Writing Commands to a Log File

The startup option `-log` is a tool that creates a log file of `rcadmin` commands that can be used by any scripts you create.

To start `rcadmin` and create a log file of the commands you use in the session, type:

```
rcadmin -log log_file_name
```

The commands you enter during the session are saved to the file. Keep in mind that all command-line errors are also written to the file. You should edit any errors out of the file after the `rcadmin` session is over. You cannot append to a log file by running the command multiple times.

To add comments to the log file during the `rcadmin` session, type:

```
#yourcomments
```

To start `rcadmin` using these saved commands, type:

```
rcadmin -input log_file_name
```

If the `login` command is included in a log session, for security reasons, the password is not saved to the file. Instead, the text string `#password here` is inserted in the file. You must replace the text string with your password. For example, the following log file changes the state of the index "collection1" to offline. You would replace *`#password_here`* with your password:

```
login
administrator
#password_here
indexstateset
server1
c
collection1
0
y
quit
```

To avoid exposing passwords in your scripts, you can use an `admincred.dat` file to store the username, password, and domain name. See "Logging in Automatically" on page 24.

Using Existing Scripts with the Current Version of rcadmin

If you have created scripts using earlier versions of `rcadmin`, and you want to use the most current version of `rcadmin`, you must update your scripts. See "Revision History" on page 30 for a description of the changes in `rcadmin`.

Note It is recommended that you update your scripts to the most current version of `rcadmin` whenever possible.

Using the version Command

If you do not need the new functionality in a new release, you can use the `version` command to filter the commands and parameters that are made available in the program.

To use this command type:

`rcadmin>version` *`version_number`*

where *version_number* can be any of the following options:

`4.5`	This emulates `rcadmin` version 4.5.x.
`5.0`	This emulates `rcadmin` version 5.0.x.
`5.5`	This emulates `rcadmin` version 5.5.
`6.0`	This emulates rcadmin version 6.0
`6.1`	This emulates `rcadmin` version 6.1

If a version number is not entered, then the most current version is emulated.

For example, typing `version 5.5` at the `rcadmin` command prompt exposes version 5.5 `rcadmin` commands.

Note If you use the `version` command, the only change you must make to existing scripts is to add the following to the command input file:

`version` *version_number*

Revision History

This section describes the changes made to `rcadmin` since K2 version 5.0.

Revisions in K2 Version 6.1

You can now use the commands `serverset`, `brokerset` and `ticketset` to change port numbers. (90500, 91926)

Revisions in K2 Version 6.0

This section describes the changes made to `rcadmin` since K2 version 5.5.

New Commands

The following commands were added:

Command	Description
`licenseset`	Changes a license key assigned to an Administration Server. See "licenseset" on page 41.
`licenseget`	Gets information about a specified license key, or a license key assigned to a specified Administration Server. See "licenseget" on page 42.
`topicsetcollattach`	Attaches a topic set to a collection. See "topicsetcollattach" on page 227.
`topicsetcolldetach`	Detaches a topic set from a collection. See "topicsetcolldetach" on page 227.

Modified Commands

The following commands were modified:

Command	Description
`brokerget`	The following prompt was added: `Internal Service:`
`indexerget`	The following prompt was added: `Internal Service:`
`paraset`	The option "`tree`" was removed from the `Index type` prompt. `Index Type [(c)ollection, (n)one]:`
`paraget`	The following prompt was added: `Internal index:`
`profileset`	The prompt `Topic Set Path` was deprecated. The topic set associated with a Profile Service is defined by the topic set's alias in the `Topic Set Alias` prompt.
`profileget`	The field `Topic Set Path` was deprecated. The topic set associated with a Profile Service is defined by the topic set's alias and is displayed in the `Topic Set Alias` field.
`searchcacheset`	The following prompt was removed: `Max Number of Searches`

`serverset/`
`serverget`
	The following prompt was added to the `serverset` and `serverget` commands:
	`Max. Active Collection Requests.`
	The following prompt was added to the `serverget` command:
	`Internal Service:`

`styleset`	The following prompts were removed:
	`Style State`
	`Style Type`
	`Admin Alias`
	`Clone Alias`
	`Path`

Command	Description

`ticketlmset/ ticketlmget`	The following prompts were removed from these commands and added to the `ticketset/ticketget` commands:
	`Enable Pre-Authentication? (y|n) (No):`
	`Client Method (0):`
	`External Authentication Data:`
	`Name():`
	`Value():`
	`User Defined():`
	
	The following prompt was added for LDAP configuration:
	`Preauth Mapping DN Key:`

`ticketset/ ticketget`	The following prompts were added to these commands:
	`Enable Pre-Authentication? (y|n) (No):`
	`Client Method (0):`
	`External Authentication Data:`
	`Name():`
	`Value():`
	`User Defined():`

Revisions in K2 Version 5.5

This section describes the changes made to `rcadmin` since K2 version 5.0.

New Commands

The following commands were added:

Command	Description
`collcreate`	creates a collection.
`collpurge`	purges all the records from a collection.
`jobcreate`	creates a new index job.
`jobmodify`	modifies an existing index job.
`jobgetdetail`	gets index job configuration information.
`jobgetstats`	gets statistics on an index job.
`jobstart`	starts an index job.
`jobstop`	stops an index job.
`jobpause`	pauses an index job.
`jobpurge`	deletes document records from a job and from the job's associated collection(s).
`jobdel`	removes an index job.
`jobresume`	resumes an index job (after paused).
`jobaddindex`	adds collections to an index job.
`jobremoveindex`	removes collections from an index job.
`styleset`	registers a styleset with a Master Administration Server.
`styleget`	gets information about available stylesets.
`styledel`	deletes registration information for a styleset.

Modified Commands

The following commands were modified:

Command	Arguments Added/Modified	Description
`adminlaunch`	`Launch as desktop application (Windows only)? [y|n]`	Prompt was added.
`collget`	`RE Settings:`	Fields in this section were removed.
`indexdel`	`Remove the index directory? [y/n]`	Prompt was added.
`indexserverget`	`Configured State`	Field was added.

`login`
`extbrokerattachset`
`extticketattachset`
`extsvcticketattachset`
`extbrokerdetach`
`extticketdetach`
`extsvcticketdetach`
	`Domain`	Prompt was changed to
		`Windows NT Domain`

`profileset`	`Style Count`
	`Alias`
	`Gateway`
		Prompts were removed.

	`Enter a Styleset alias for the following Gateways (press Enter for none):`
	`File System`
	`HTTP`
	`Lotus Notes`
	`Documentum`
	`ODBC`
	`Exchange`
		Prompts were added.

`querylogset/` `querylogget`	`Query Suggestions? [y|n]` `Query Suggestion Time? [y|n]`	Prompts were added.
`reset/reget`	`User Index Path`	Prompt was removed. The path to a user index is set using the `riset` command, and is shown using the `riget` command.
	`Map Terms`	Prompt was changed to `Profile Decryption`.
`riset`	`Type`	Prompt was changed to `Recommendation Type`.

Command	Arguments Added/Modified	Description
`serverget`	`User Recommendation Enable`	Prompt was removed.
`ticketset/ticketget`	`Enable External USER_PARAM keys from Login Credentials [y|n]`	Prompt was added for pre-authentication support.

`ticketlmset/ ticketlmget`	`Limit Trusted Domain Enumeration:`
	`Domain Count (override default)(0):`
	`Domain(i/n)`
	`Name():`
	`Controller():`
	`User Group Enumeration Type:`
	`User Group Authentication Type:`
	`Enable Pre-Authentication? [y|n]`
	`Client Method (0):`
	`External Authentication Data:`
	`Name():`
	`Value():`
	`User Defined():`
	`Enable Built In Groups? [y|n]`
	`Custom Built In Group Count (override default)(0):`
		Prompts were added for pre-authentication support.

Deprecated Commands

The following commands were deprecated:

Deprecated Command	Alternative Command
`collreset`	Use the `riset` command to create `doc` type recommendation indexes. See "riset" on page 242.
`reoptionset`	Use the `reset` command to set host-wide recommendation defaults, such as Score Factor. See "reset" on page 237.
`reoptionget`	Use the `reget` command to view host-wide recommendation defaults, such as Score Factor. See "riget" on page 244.
`reuserset`	Use the `riset` command to create `user` type recommendation indexes. See "riset" on page 242.

Note These commands were deprecated in the K2 Administration API, and therefore are not available with the `-version` command. See "Using the version Command" on page 28.

2

K2 System Commands

This chapter describes the `rcadmin` commands that help you manage the K2 system.

- Viewing Services and Indexes in the K2 System
- Managing Licenses
- Managing K2 Domain Names

Viewing Services and Indexes in the K2 System

The commands in this section show K2 system administration information in three different ways: service hierarchy view, host view, and index view.

For more information about the output shown by each of these command, see the appropriate "get" commands for each K2 service or index (for example, `brokerget` for K2 Broker information, and `collget` for collection information).

This section describes the following commands:

- hierarchyview
- hostview
- indexview

hierarchyview

The `hierarchyview` command shows all K2 services in the context of the entire K2 system. For example, it shows relationships between K2 Brokers and K2 Servers, even if these services and their attached indexes are located on machines in different cities or on different continents. This command does not group K2 components according to physical boundaries. To determine which services reside in a specific physical location, use the `hostview` command. For external services, the "Configured State" is always shown as "not available" since K2 cannot tell what state the service is in.

Input

Enter the following information when prompted:

Argument	Description
`Service Type`	The type of services whose information you want to view. The following options are available: `a` = K2 Administration Servers. `b` = K2 Brokers. `s` = K2 Servers. `t` = K2 Ticket Servers. `i` = K2 Index Servers. To show information about all types of services, press the `Enter` key.

hostview

Like the `hierarchyview` command, the Host view shows all services and indexes in the K2 system. However, it groups these services according to their physical location on host computers. The Host view also shows jobs.

Input

Enter the following information when prompted:

Argument	Description
`Admin Service Alias [optional]`	The unique identifier for an Administration Server. `rcadmin` shows information about all hosts controlled by the specified Administration Server. To show information about all hosts, press the `Enter` key.

indexview

The `indexview` command shows all the indexes that store content, grouping them by collection, knowledge tree, topic set, parametric index, and recommendation index.

Input

Enter the following information when prompted:

Argument	Description
`Index Type`	The type of indexes whose information you want to view. The following options are available: `a` = all indexes. `c` = collections. `t` = knowledge trees. `p` = parametric indexes. `s` = topic sets. `r` = recommendation indexes.
`Index Alias`	The unique identifier for the index whose information you want to view. This is optional.

Managing Licenses

The commands in this section enable you to set a new license key for an Administration Server, view information about Verity licenses, refresh the list of licenses, and control CPU binding configuration.

This section describes the following commands:

- licenseset
- licenseget
- licensecheck
- licenseservice
- licenseupdate
- servercpuset

licenseset

The `licenseset` command enables you to change the license key assigned to an Administration Server. By default, a local Administration Server inherits the K2 Domain License Key defined in the Master Administration Server. You can use this command to define an alternative "private" license key for the local host. For example, you can set a new license key that includes a trial version of a previously disabled feature, and later set another license key to include a full version of the feature.

IMPORTANT When you set a new license key, ensure you do it before or after peak indexing activity.

While trial licenses are useful for evaluating K2 features in a test environment, they are not recommended for production environments.

After changing a license key, you must restart all services on the associated host. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server whose license you want to set.
`License Key`	The new license key.
`Organization Name`	The organization name.

Note If you need to restore your former license key, K2 saves a backup of its two associated files:

`vdk30.lic.bak`

`runtime.lic.bak`

These are stored at *`install`*`\k2_6\k2\common`.

If you make several consecutive license key changes, only the most recent is captured in the backup data.

Example

```
rcadmin> licenseset
Admin Alias:admin1
License Key (XXXX-XXXX-XXXX-XXXX):2RIGHR3-3SN4CK5-3TDVMZM-9VPP7NB
Organization Name:Verity, Inc.
<<Return>> SUCCESS
```

licenseget

The `licenseget` command shows information about a specified license key, or a license key assigned to a specified Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server whose license information you want to view. This is required to retrieve licensing information for a specific Administration Server, and to retrieve licensing information for any license key not assigned to the Administration Server.
`License Key`	To retrieve information about a specific license key, enter the license key. To retrieve information about the license key assigned to the specified Administration Server, press the `Enter` key at this prompt. The `licenseget` command retrieves information about license keys for K2 V6.0 and higher. If you request information on a license for another K2 version, the error message "`invalid license`" is shown.
`Organization Name`	The organization name for the specified license key.

Output

For each feature included in the license, `rcadmin` shows the following information:

Argument	Description
`Alias:`	The unique identifier for the K2 feature. The aliases are created by the K2 installer. For example, `exchange`, for the Exchange Gateway.
`Description:`	A text description of the feature.
`Information:`	The version number for particular features included in the license. It also lists the number of CPUs for which the K2 Broker or Server features are licensed. When a feature is associated with a trial license, this field lists the number of days after which that feature expires.
`Type:`	The feature type. The following options are available: `Unknown` = feature type is unknown. `Service` = feature is a K2 service. `Gateway` = feature is a Verity gateway. `Application` = feature is a Verity application. `Locale` = feature is a Verity locale.
`License State:`	The state of the feature's license. The following options are available: `None` = feature is not license-controlled. `Invalid` = license is invalid. `Valid` = license is valid. `Trial` = license is for a trial version. `Expired` = license has expired. `Exceeded` = number of features installed exceeds the number of features licensed. `Unknown` = state is unknown.
`Install State:`	The installation state of the feature. The following options are available: `Installed` = feature is installed. `Uninstalled` = feature is not installed. `Unknown` = state is unknown.

Example

```
rcadmin> licenseget
Admin Alias:admin1
License Key (XXXX-XXXX-XXXX-XXXX):2RIGHR3-3SN4CK5-3TDVMZM-9VPP7NB
Organization Name:Verity, Inc.
License Information(1/67):
  Alias        : k2search
  Description  : K2 Server
  Information  : 99 CPUs; Version 6.1.0
  Type         : Service
  License State: None
  Install State: Installed

License Information(2/67):
  Alias        : k2reporting
  Description  : K2 Reporting
  Information  : Version 6.1.0
  Type         : Service
  License State: None
  Install State: Installed

License Information(3/67):
  Alias        : k2ticket
  Description  : K2 Ticket
  Information  : Version 6.1.0
  Type         : Service
  License State: None
  Install State: Installed

License Information(4/67):
  Alias        : viewing
  Description  : K2 Viewing Server
  Information  :
  Type         : Service
  License State: None
  Install State: Installed
...
```

licensecheck

The `licensecheck` command shows information for a specified feature, a specified feature type, a specified feature state or all features on an Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`License Alias [leave blank to list by type]`	The unique identifier for the feature whose license information you want to view. This is optional. The aliases are created by the K2 installer. or example, `exchange`, for the Exchange Gateway. To show information about *all* features, press the `Enter` key at this prompt, type `0` (zero) at the `License Type` prompt, and type `2` at the `License Query Type` prompt.
`License Type`	The feature type for which to show license information. The following options are available: `0` = all feature types. `1` = K2 services. `2` = Verity gateways. `3` = Verity applications. `4` = Verity locales.
`License Query Type`	Indicates whether license information about unavailable features, and/or available features is displayed. The following options are available: `0` = Unavailable. Show information about features that are not installed and/or not licensed. `1` = Available. Show information about features that are installed and licensed (if required). `2` = Any. Show information about unavailable and available features.

Output

For each feature, `rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the feature. The aliases are created by the K2 installer. or example, `exchange`, for the Exchange Gateway.
`Description`	A text description of the feature.
`Information`	The version number for particular features included in the license. It also lists the number of CPUs for which the K2 Broker or Server features are licensed. When a feature is associated with a trial license, this field lists the number of days after which that feature expires.
`Type`	The feature type. The following options are available: `Unknown` = feature type is unknown. `Service` = feature is a K2 service. `Gateway` = feature is a Verity gateway. `Application` = feature is a Verity application. `Locale` = feature is a Verity locale.
`License State`	The state of the feature's license. The following options are available: `None` = feature is not license-controlled. `Invalid` = license is invalid. `Valid` = license is valid. `Trial` = license is for a trial version. `Expired` = license has expired. `Exceeded` = number of features installed exceeds the number of features licensed. `Unknown` = state is unknown.
`Install State`	The installation state of the feature. The following options are available: `Installed` = feature is installed. `Uninstalled` = feature is not installed. `Unknown` = state is unknown.

licenseservice

The `licenseservice` command shows the host machine on which a specified feature is available.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`License Alias`	The unique identifier for the feature. The aliases are created by the K2 installer. or example, `exchange`, for the Exchange Gateway. You can determine the names of license aliases by using the `licensecheck` or `licenseget` command.

Output

For the feature, `rcadmin` shows the following information:

Argument	Description
`Host`	The host on which the specified feature is available.

licenseupdate

The `licenseupdate` command refreshes the list of licenses on a specified Administration Server. Normally, the K2 installer automatically updates the license list whenever a component is installed.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.

Output

`rcadmin` shows information that may be useful to Verity Technical Support if you need to diagnose licensing problems.

servercpuset

The `servercpuset` command enables you to change the CPU binding configuration for a K2 Server. This is useful when you purchase additional CPU licenses after your initial installation, or when you have multiple servers on the same machine and you need to distribute CPUs between them.

Note On Linux, CPU binding is only supported on Red Hat Enterprise Linux AS 3 and higher, and SUSE Linux Enterprise Server 9 and higher.

Note After changing the CPU binding configuration, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

See the *Verity K2 Dashboard Administrator Guide* for more information on CPU binding.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 Server.
`Start Processor ID`	If you have more CPUs on the host than are covered by your K2 license, enter the ID of the first CPU that the K2 Server should use. Valid values are zero to 99. For example, if your four-CPU host has physical processor IDs 6, 9, 10, and 11, and you have a two-CPU K2 license, you could simply specify any of the four CPUs on the host (0, 1, 2 or 3). You do not have to specify your actual processor IDs. In this example, if you specified 0 as the starting point, K2 interprets this as the first and second CPUs, or IDs 6 and 9.
`Number of Processors`	If you have more CPUs on this host than are covered by your K2 license, enter the total number of CPUs that the K2 Server should use. In the event you purchased additional CPU licenses after your initial installation, you need to reconfigure the binding. In this case, you would increase the value in this field to reflect the total number of licensed CPUs. You may also want to adjust the `Start Processor ID` value to reflect the new physical processors.

Managing K2 Domain Names

The commands in this section manage the names of local K2 domains. For more information on support for multiple K2 domains, see the *Verity K2 Installation and Setup Guide* and "Managing External K2 Services" on page 90.

This section describes the following commands:

- hostdomainset
- hostdomainget

hostdomainset

The `hostdomainset` command creates, modifies, or removes the name of the local K2 domain.

Note After naming the domain, perform a full restart on the Master Administration Server, and the local Administration Servers connected to the Master Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type (Update=0, Remove=1)`	Type `0` to specify or change the name of a local K2 domain. Type `1` to remove the name of a local K2 domain.
`Domain`	The name of the local K2 domain. The name cannot contain '.' (period), '@', or '/'.
`Description`	An optional text description of the local K2 domain.
`Options`	Reserved for future use. Press the `Enter` key.

hostdomainget

The `hostdomainget` command shows the name of the local K2 domain.

For more information on support for multiple K2 domains, see the *Verity K2 Installation and Setup Guide* and "Managing External K2 Services" on page 90.

Output

`rcadmin` shows the following information:

Argument	Description
`Domain`	The name of the local K2 domain.
`Description`	An optional text description of the local K2 domain.
`Options`	Reserved for future use.
`Version`	The version of the Master Administration Server or Administration Server on that host.
`Build`	The build number of the Master Administration Server or Administration Server on that host.

Example

```
rcadmin> hostdomainget
K2 Domain Information:
    Domain          : verity
    Description     : The Verity domain
    Options         : 0
    Version         : 60
    Build           : Build 20031125
```

3

Managing K2 Services

This chapter describes the commands used to manage K2 services, such as Administration Servers, K2 Brokers, K2 Servers, K2 Ticket Servers, and K2 Index Servers.

- Adding a K2 Service
- Managing K2 Administration Servers
- Managing K2 Brokers
- Managing K2 Servers
- Managing K2 Ticket Servers
- Managing K2 Index Servers
- Managing External K2 Services
- Attaching and Detaching Services
- Starting and Stopping K2 Services
- Checking the State of a K2 Service
- Removing K2 Services
- Managing Recommendation Transaction Logs
- Configuring the Search Results Cache

Adding a K2 Service

To add a K2 service to a K2 system, follow these steps:

1. Add the K2 service to the Administration Server using the appropriate "set" command for the service. For example, `serverset`, `brokerset`, `ticketset`, and `indexerset`.

2. Associate an index with the K2 Server or K2 Index Server.

   When you add a K2 Server or K2 Index Server to your system, there is no content associated with it. To associate an index with a K2 Server, you must first register, *and* then attach the index. To associate an index with a K2 Index Server, you must only register the index. Unlike a K2 Server, an index does not have to be attached to a K2 Index Server before it can be used.

   When you register an index, you provide an alias for the index and give its location for the host computer. The path you specify must be correct for the host machine. You register an index only once per host.

   For more information, see "Registering Indexes" on page 190, and "indexattach" on page 194.

3. If required, attach the K2 service to an Administration Server, K2 Broker or K2 Server. See "Attaching and Detaching Services" on page 104.

4. Start the K2 service using the `servicesignal` command. This also adds the service to the watched services list. An Administration Server monitors watched services, and starts and stops them as necessary. See "servicesignal" on page 119.

Use the `servicedel` command to remove a K2 service from a host. See "Removing K2 Services" on page 127.

Use the `hostview` command to show the services and indexes that are defined for a particular host. See "Viewing Services and Indexes in the K2 System" on page 38.

Configuring Service Logs

K2 Administration can generate service logs for a particular service. Service logs are a valuable diagnostic tool, often used by Verity Technical Support to determine how your K2 system can work most efficiently. K2 Administration offers you the following logs for viewing:

- status logs for K2 Brokers, K2 Servers, K2 Ticket Servers, K2 Index Servers, and Administration Servers
- query logs for K2 Brokers and K2 Servers

The status log records various diagnostic messages on the status of a K2 service. To configure the type of messages recorded in the status log, use the `statuslogset` command. See "Managing Status Logs" on page 148 for more information.

A query log records user query activity on a K2 Broker or K2 Server. To configure which user query information is recorded in the log, use the `querylogset` and `querylogexcludeset` commands. See "Managing Query Logs" on page 152.

Using Trusted Clients to Control Connections

The *trusted client* feature enables you to control which computers (IP addresses) can connect to a particular service. By default, K2 services accept communications from any IP address, although, a login may be required to perform some or all operations.

If you use trusted clients, you must specify IP addresses for all computers that connect to the service, including K2 clients, Administration Servers, K2 Brokers, K2 Servers, K2 Ticket Servers, and K2 Index Servers. If you do not, the excluded services will not connect to your K2 system, and the excluded K2 clients will be unable to access it.

To manage trusted clients, use the following commands:

- trustedclientset
- trustedclientget
- trustedclientdel

See "Managing Trusted Clients" on page 164 for more information.

Managing K2 Administration Servers

The commands in this section manage K2 Administration Servers. A K2 *Administration Server* is a repository for configuration information, and monitors K2 components. See "Introducing Administration Servers" on page 18 for more information.

This section describes the following commands:

- adminset
- adminget

adminset

The `adminset` command enables you to modify some properties of an existing K2 Administration Server. For example, you can use this command to change the contact email address for K2 notification. K2 sends messages to this address if it detects serious problems that require immediate attention.

Note Do not use the `adminset` command to *add* a new host to the system because the command cannot create all the data necessary for a new host to operate. To add a new host, use the K2 installer.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for an Administration Server, or insert information for a new Administration Server. The following options are available: `0` = Update existing information for an Administration Server. `1` = Insert a new Administration Server. NOTE: Only use *Insert* mode if Verity Technical Support advises you to do so. Normally, you use the K2 installer to add a new host.
`Admin Alias`	The unique identifier for the Administration Server.
`Port`	The port used by the Administration Server. NOTE: This value cannot be changed for an existing host.

Argument	Description
`Admin Type`	The type of server. Type `1` if it is a Master Administration Server, or `2` if it is a non-Master Administration Server. You only see this prompt if you are modifying an Administration Server because you cannot add a Master Administration Server using this command. A slave Administration Server is added by default. NOTE: Do not change this setting for an existing host.
`Description`	A text description of the Administration Server.
`VDK Home`	This value is not used.
`Install path`	This value is not used.
`CharMap`	This value is not used.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.
`Host Name`	The unique identifier for the machine hosting the Administration Server.
`Email Notification`	The notification email address used by K2. K2 sends messages to this address if it detects serious problems that require immediate attention, such as a failure to start up the Master Administration Server, or when a K2 service is restarted by K2 Administration. K2 provides two classes of email notification: Admin-1 (related to K2 services) and Admin-2 (related to jobs). For information on setting an email notification level, see the *Verity K2 Dashboard Administrator Guide*.
`Number of Listeners`	The number of listener threads used by this Administration Server. This controls the number of concurrent connections to the server. The default is 100. If you change the value in this field, perform a full restart on the Administration Server using the `adminsignal` command. This applies the change to the system. See "adminsignal" on page 117

adminget

The `adminget` command enables you to view information about a K2 Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the Administration Server.
`Port`	The port used by the Administration Server.
`Server Spec`	The specification for the machine hosting this Administration Server.
`Description`	A text description of the Administration Server.
`Type`	Indicates whether the Administration Server is a Master Administration Server or a non-Master Administration Server.
`CharMap`	This value is not used.
`VDK Home`	This value is not used.
`Install path`	This value is not used.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful when you have more than one network card and want to only accept requests on one of them.
`Host Name`	The unique identifier for the machine hosting the Administration Server.

Argument	Description
`Email`	The notification email address used by K2. K2 sends messages to this address if it detects serious problems. For more information, see the *Verity K2 Dashboard Administrator Guide*.
`Listeners`	The number of listener threads used by this Administration Server. This controls the number of concurrent connections to the server.
`Active XML File`	The name of the active `admin`*N*`.xml` file.

Managing K2 Brokers

The commands in this section manage K2 Brokers. A *K2 Broker* is a K2 service that receives client search requests and distributes them to available K2 Servers.

After adding a K2 Broker to the system, it can be attached to another K2 Broker. See "Attaching and Detaching Services" on page 104.

To start a K2 Broker, use the `servicesignal` command. See "servicesignal" on page 119. A restart using this command also adds the service as a watched service. K2 automatically restarts a watched service if it finds it stopped.

This section describes the following commands:

- brokerset
- brokerget

brokerset

The `brokerset` command enables you to add a K2 Broker to an Administration Server, or modify information for an existing K2 Broker.

Input Parameters

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for a K2 Broker, or add a new K2 Broker. The following options are available: `0` = Update existing information for a K2 Broker. `1` = Insert a new K2 Broker.
`Broker Alias`	The unique identifier for the K2 Broker. The K2 Broker alias should include the host name to ensure that each server has a unique name in a multi-host K2 domain. For example, an appropriate name would be `myhost_broker1`.
`Port`	The port used by the K2 Broker. If you change this value, perform a full restart on the K2 Broker using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Description`	A text description of the K2 Broker.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful when you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Broker is attached. In a multi-host system, this is any Administration Server alias. In a single-host system, this is the Master Administration Server alias for this machine. This is normally the machine's host name. NOTE: This setting cannot be changed for an existing K2 Broker.
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	This value is not currently used.

Argument	Description
`Max. Listener Threads`	Controls how many clients and other K2 services can connect to this K2 Broker, such as K2 Servers that serve content from this service, as well as the K2 Admin process that monitors the state of this service. The clients that connect can be C applications such as `rck2`, or Java/COM clients, such as the Verity templates. This value is considered a maximum because K2 manages the number of listeners—depending on the load—up to this value. The default is 200. Since a UNIX system has a limited number of file descriptors per process, it is recommended the maximum number of listeners not exceed 256 on UNIX. If you change this value, perform a full restart on the K2 Broker using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Ping Delay (msecs`	The frequency with which the K2 Broker pings the K2 Broker to which it is attached. The ping informs the upper-level broker of the lower-level broker's current status. When the ping is received, the upper-level broker connects to the lower-level broker and updates its configuration information. The ping delay is specified in milliseconds. The default and minimum value is 300000 milliseconds (5 minutes). If a value smaller than this is entered, 300000 is used.
`Limit Replication?`	This applies to a K2 Broker attached to two or more K2 Servers that contain a mirrored collection. If you set this value to `y`, all information for a query using the mirrored collection is taken from the same K2 Server. This avoids problems if the mirrored collections are accidentally out of synchronization. If you set the value to `n`, the K2 Broker uses any available K2 Server.
`Connection Timeout (msecs)`	The length of time to wait, in milliseconds, for a response before functions time out.
`Max. Search Time (msecs)`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.
`Enable Queue Polling?`	Indicates whether the queue polling is enabled to remove expired search requests from the search queues. By default, this is enabled.

brokerget

The `brokerget` command enables you to view information about a K2 Broker.

Input

Enter the following information when prompted:

Argument	Description
`Broker Alias`	The unique identifier for the K2 Broker whose information you want to view.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the K2 Broker.
`Port`	The port used by the K2 Broker.
`ServerSpec`	The specification for the machine hosting the K2 Broker.
`Description`	A text description of the K2 Broker.
`Configured State`	The state of the K2 Broker according to the `admin`*N*`.xml` file. The following options are available: `offline` = offline and is not available. `online` = online and is available. `hidden` = hidden. It is not available to users, but is available to administrators.
`Internal Service`	Indicates whether the service is reserved for internal K2 functionality. For example, an Index Server created for use by the Verity Business Console is an internal service.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Broker is attached.

Argument	Description
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	This value is not currently used.
`Max. Listener Threads`	The maximum number of clients and other K2 services that can connect to this K2 Broker, such as K2 Servers that serve content from this service, as well as the K2 Admin process that monitors the state of this service. The clients that connect can be C applications such as `rck2`, or Java/COM clients, such as the Verity templates.
`Ping Delay`	The frequency with which the K2 Broker pings the K2 Broker to which it is attached. The ping informs the upper-level broker of the lower-level broker's current status. When the ping is received, the upper-level broker connects to the lower-level broker and updates its configuration information. The ping delay is specified in milliseconds. The default and minimum value is 300000 milliseconds (5 minutes). If a value smaller than this is entered, 300000 is used.
`Limit Replication`	This applies to a K2 Broker attached to two or more K2 Servers that contain a mirrored collection. If you set this value to `y`, all information for a query using the mirrored collection is taken from the same K2 Server. This avoids problems if the mirrored collections are accidentally out of synchronization. If you set the value to `n`, the K2 Broker uses any available K2 Server.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.
`Max. Search Time`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.
`Enable Queue Polling`	Indicates whether the queue polling is enabled to remove expired search requests from the search queues. By default, this is enabled.
`Search Cache:`	
`Enable`	Indicates whether the K2 Broker caches search results. Caching results can improve the performance of your K2 system.

Argument	Description
`Timeout`	The length of time to wait, in milliseconds, for a response before the search cache times out.
`Max Number of searches`	The maximum number of search results cached.
`Max Memory Size Used by Search`	The maximum size of the result cache in bytes.
`Transaction Log:`	This section contains information about the K2 Broker's transaction log. This is a log of the transactions between the K2 Broker and its K2 Servers. A transaction is a modification of one or more entities in a recommendation index. For example, a transaction may make a document more relevant to a particular query due to user input. See "retxlogset" on page 129.
`Enable`	Indicates whether K2 Broker/Server transactions are logged to the XML file. The following options are available: `0` = Transaction logging is disabled. `1` = Transaction logging is enabled. The K2 Broker can receive requests from a *VTransaction* API call, and log the information in an XML file. The default path of the XML file is *`dataDir`*`/host/`*`brokeralias`*`/re`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). The file name is `tr.xml`. The path can be configured through the `retxlogset` command. See "retxlogset" on page 129. The VTransaction API provides an interface for sending Transactional Updates to the Recommendation Engine. An update action allows the profile of a user or document to be updated by a query, document, user, or set of raw terms.
`Log path`	The directory path to the transaction log.
`Max Transactions Per Hour`	Indicates the rate of transaction logging, that is, the rate at which the K2 Broker samples VTransaction requests. The sampling of requests is a useful feature in cases when the K2 Broker can receive a lot of requests, and it suffices to select and log only a subset of these requests. By default, the K2 Broker records every request. If this parameter is set to a value such as 60, then the K2 Broker samples incoming requests at the rate of 60/hour, or 1/minute. If, in the same minute, 2 requests arrive at the K2 Broker, then the second request is skipped (or ignored). To avoid this situation, set the rate to a high value, such as 10000, or do not set it at all.

Managing K2 Servers

The commands in this section manage K2 Servers. A *K2 Server* is a K2 service that receives search, viewing, profiling, and recommendation requests and performs searches of collections, knowledge trees, parametric indexes, RE Doc Indexes, and RE User Indexes.

After adding a K2 Server to the system, it must be attached to a K2 Broker. See "Attaching and Detaching Services" on page 104.

To start a K2 Server, use the `servicesignal` command. See "servicesignal" on page 119. A restart using this command also adds the service as a watched service. K2 automatically restarts a watched service if it finds it stopped.

When you add a K2 Server, there is no content associated with it. To associate a collection, parametric index, or knowledge tree with a K2 Server, you must first register, and then attach these indexes. For more information, see Chapter 7.

Note For information on K2 Report Servers and K2 Reporting, see the *Verity K2 Dashboard Administrator Guide*. Verity K2 Reporting is a service that allows the K2 Dashboard or an application to provide reports on user activity. Through K2 Reporting, administrators can generate, view, and export reports that summarize search activity.

This section describes the following commands:

- serverset
- serverget
- vdksettingset

serverset

The `serverset` command enables you to add a K2 Server to an Administration Server, or modify information for an existing K2 Server.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for a K2 Server, or add a new K2 Server. The following options are available: `0` = Update existing information for a K2 Server. `1` = Insert a new K2 Server.
`Server Alias`	The unique identifier for the K2 Server. The K2 Server alias should include the host name to ensure that each server has a unique name in a multi-host K2 domain. For example, an appropriate name would be `myhost_server1`.
`Port`	The port used by the K2 Server. If you change this value, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Description`	A text description of the K2 Server.
`Server Type:`	This section indicates the types of requests handled by the K2 Server. By default, a K2 Server handles search, viewer and profiler requests. Optionally, you can designate some K2 Servers to only handle viewing requests, and other K2 Servers to only handle search requests. Since viewing documents is more CPU-intensive than searching, this enables you to make sure that viewing requests do not slow down the searches. For more information, see the *Verity K2 Getting Started Guide*.
`Search Server?`	Press `Enter` if the K2 Server handles search requests.
`Viewer?`	Press `Enter` if the K2 Server handles viewing requests.
`Profiler?`	Press `Enter` if the K2 Server handles Profiler requests.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Server is attached. In a multi-host system, this is any Administration Server alias. In a single-host system, this is the Master Administration Server alias for this machine. This is normally the machine's host name. This setting cannot be changed for an existing K2 Server.
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	This value is not currently used.
`Max. Listener Threads`	Controls how many clients and other K2 services can connect to the K2 Server, such as K2 Brokers that serve content from this service, as well as the K2 Admin process that monitors the state of this service. The clients that connect can be C applications such as `rck2`, or Java/COM clients, such as the Verity templates. This value is considered a maximum because K2 manages the number of listeners—depending on the load—up to this value. The default is 200. Since a UNIX system has a limited number of file descriptors per process, it is recommended the maximum number of listeners not exceed 256 on UNIX. If you change this value, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Number of Threads`	The default number of threads used by each service (for example, collection search) in this K2 Server. If you change this value, perform a full restart on the K2 Server using the `servicesignal` command (or the K2 Dashboard). This applies the change to the system. See "servicesignal" on page 119.
`Server VDK Sorting?`	If you enable this, VDK sorts based on string fields when it delivers results. By default, VDK sorts documents based on scores only, which makes searches faster.
`Ping Delay`	The frequency with which the K2 Server pings the K2 Broker to which it is attached. The ping informs the Broker of the Server's current status. When the ping is received, the Broker connects to the Server and updates the Server's configuration information. The ping delay is specified in milliseconds. The default and minimum value is 300000 milliseconds (5 minutes). If a value smaller than this is entered, 300000 is used.

Argument	Description
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.
`Max. Search Time (msecs)`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.
`Enable Queue Polling?`	Indicates whether the queue polling is enabled to remove expired search requests from the search queues. By default, this is enabled.
`Max. Active Collection Requests`	The maximum number of collection search requests that can be processed simultaneously on a CPU for a K2 Server. If the number of requests surpasses this value, the overloading request is not executed until the number of requests falls below this value. This ensures the total number of active threads on the K2 Server remains constant, and consequently, improves performance. The default is 5. The range is 4 to 10. The final value for the K2 Server is calculated by multiplying this field by the total number of CPUs on which the K2 Server is running. If the K2 Server is running on a slow machine and/or the CPU resources are shared with other memory-intensive applications, you may need to lower this value. If the K2 Server is running on a fast machine and operating system, you may need to increase this value. If you change this value, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.' For more information, see the *Verity K2 Dashboard Administrator Guide*.

serverget

The `serverget` command enables you to view information about K2 Servers.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the K2 Server.
`Port`	The port used by the K2 Server.
`ServerSpec`	The specification for the machine hosting the K2 Server.
`Description`	A text description of the K2 Server.
`Server Type`	The types of requests handled by the K2 Server. Possible types are `Search`, `Viewer`, and `Profiler`. By default, a K2 Server handles all types of requests. You can designate some K2 Servers to handle only viewing requests, other K2 Servers to only handle search requests, and other K2 Servers to handle profiler requests. Since viewing documents is more CPU-intensive than searching, this enables you to make sure that viewing requests do not slow down the searches.
`Configured State`	The state of the K2 Server according to the `admin`*N*`.xml` file. `offline` = The K2 Server is offline and is not available. `online` = The K2 Server is online and is available. `hidden` = The K2 Server is hidden. It is not available to users, but can be operated on by administrators.
`Internal Service`	Indicates whether the service is reserved for internal K2 functionality. For example, an Index Server created for use by the Verity Business Console is an internal service.

Argument	Description
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which this K2 Server is attached.
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	This value is not currently used.
`Max. Listener Threads`	The maximum number of clients and other K2 services that can connect to the K2 Server, such as K2 Brokers that serve content from this service, as well as the K2 Admin process that monitors the state of this service. The clients that connect can be C applications such as `rck2`, or Java/COM clients, such as the Verity templates.
`Threads`	The default number of threads used by each service (for example, collection search) in this K2 Server.
`VDK Sorting`	Indicates whether K2 sorts at the collection level. If you enable this, VDK sorts based on string fields when it delivers results. By default, VDK sorts documents based on scores only, which makes searches faster.
`Ping Delay`	The frequency with which the K2 Server pings the K2 Broker to which it is attached. The ping informs the K2 Broker of the K2 Server's current status. When the ping is received, the Broker connects to the Server and updates the Server's configuration information. The ping delay is specified in milliseconds. The default and minimum value is 300000 milliseconds (5 minutes). If a value smaller than this is entered, 300000 is used.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.
`Max. Search Time`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.

Argument	Description
`Enable Queue Polling`	Indicates whether the queue polling is enabled to remove expired search requests from the search queues. By default, this is enabled.
`Max. Active Collection Requests`	The maximum number of searches that can be processed simultaneously on a K2 Server. If the number of simultaneous searches on a K2 Server surpasses this value, the overloading search request is blocked and is not executed until the number of search requests falls below this value.
`Global VDK Settings:`	Settings for the VDK kernel.
`VDK Stream File`	Indicates whether VDK is allowed to stream *any* document from the gateway (not only those in the collection). By default, this is set to `n`. Setting it to `y` has security implications because it enables the user to view more documents than usual.
`MaxDocBlockSize`	The maximum size, in bytes, of blocks to be read by K2.
`VDK Field Read:`	
`Max Column Size`	The maximum size, in bytes, for a field retrieved from a collection. Make sure this is as large as the largest field you expect to use.
`gwCertTimeOut`	The length of time to wait, in milliseconds, for a certificate from the gateway before the command times out.
`DateInputFormat`	The format for parsing dates. See the "Date Formats" appendix in the *Verity K2 Dashboard Administrator Guide*.
`KnowledgeBase`	The directory path to the knowledge base. A *knowledge base* consists of one or more topic sets that can be used as search criteria in queries. For more information on knowledge bases, see the *Verity Query Language and Topic Guide*.
`Topic Set Path`	The directory path to the topic set. A *topic set* is a group of topics that have been compiled for use by a Verity application. A *topic* is a stored query expression that is written in the Verity Query Language (VQL) and is used for classifying documents in a collection. For more information on topic sets, see the *Verity Query Language and Topic Guide*.

Argument	Description
`Locale`	The default Verity locale used by the K2 Server to open a Verity index. A *locale* is a software package that enables K2 to operate on documents in a wide variety of languages. Locales provide language-specific tokenization, stemming, recognition of parts-of-speech, recognition of noun phrases, and thesauruses. See the *Verity Locale Configuration Guide* for acceptable values.
`Charmap`	The default character set used by the K2 Server to display results for a Verity index. See the *Verity Locale Configuration Guide* for acceptable values.
`Memory Usage:`	
`Max Memory Size`	The global maximum memory (in KB) for all indexes, unless a value is specified in the particular index.
`Max Files`	The maximum number of file handles that can be opened by a specific search thread.
`Max Memory percentage`	The maximum amount of memory, in terms of a percentage, that this server can use.
`Number of 1K Pages`	The global value for the number of pages (1024 bytes) for disk caching of the collection partitions, unless a value is specified in the particular index.
`Source Query Cache:`	
`Cache Size`	The maximum size, in bytes, of the source query cache.
`Shared TFields`	Indicates whether the source query cache is enabled. The cache requires multiple VDK sessions to share TFields. See "Transitory Fields (TFields)" in the *Verity Developer's Kit Programming Reference*.
`Search Cache:`	
`Enable`	Indicates whether the K2 Server caches search results. Caching results can improve the performance of your K2 system.
`Timeout`	The length of time to wait, in milliseconds, for a response before the search cache times out.
`Max Number of Searches`	The maximum number of search results cached.

Argument	Description
`Max Memory Size Used by Search`	The maximum size of the result cache in bytes.
`Transaction Log:`	This section contains information about the K2 Server's transaction log. This is a log of the transactions between the K2 Broker and its K2 Servers. The settings are configured through the `retxlogset` command. See "retxlogset" on page 129.
`Enable`	Indicates whether K2 Broker/Server transactions are logged for document recommendation. If transaction logging is enabled, the K2 Server can receive requests from a VTransaction API call and log the information in an XML file. The default path to the XML file is *`dataDir`*`/host/serveralias/re`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). The file name is `tr.xml`. The path can be configured through the `retxlogset` command. See "retxlogset" on page 129.
`Log Path`	The directory path to the transaction log.
`Max Transactions Per Hour`	Indicates the rate of transaction logging, that is, the rate at which the K2 Server samples VTransaction requests. The sampling of requests is a useful feature in cases when the K2 Server can receive a lot of requests, and it suffices to select and log only a subset of these requests.
`Start Processor ID`	If you have more CPUs on this host than are covered by your K2 license, this is the ID of the first CPU that K2 will should. Valid values are zero to 99.
`Number of Processors`	If you have more CPUs on this host than are covered by your K2 license, this is the number of CPUs that K2 should use. The default is 0 which indicates no CPU binding is in effect.

vdksettingset

The `vdksettingset` command enables you to configure the default VDK settings for a K2 Server.

Note These settings can be overridden for individual collections and knowledge trees by using the `indexvdkset` command. See "indexvdkset" on page 198.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	`The unique identifier for the K2 Server.`
`VDK Setting:`	The prompts in this section correspond to the VDK configuration settings for the K2 Server.
`Allow Stream File?`	Indicates whether VDK is allowed to stream *any* document from the gateway (not only those in the collection). By default, this is set to `n`. Setting it to `y` has security implications because it enables the user to view more documents than usual.
`Max Doc Block Size`	The maximum size, in bytes, of blocks to be read by K2.
`VDK Field Read:`	
`Max Column Size`	The maximum size, in bytes, to read from each field. Make sure this is as large as the largest field you expect to use.
`gw Cert Timeout (msecs)`	The length of time to wait, in milliseconds, for a certificate from the gateway to expire.
`Date Input Format`	The default date format for all date inputs to K2. For more information on date formats, see the *Verity K2 Dashboard Administrator Guide*.
`Knowledge Base Path`	The directory path to the default knowledge base.
`Topic Set Path`	The directory path to the default topic set.

Argument	Description
`Locale`	The default Verity locale used by the K2 Server to open a Verity index. If a locale is not defined for an individual index, or for a Profile Service, the value in this VDK setting is used. See the *Verity Locale Configuration Guide* for acceptable values.
`Charmap`	The default character set used by the K2 Server to display results for a Verity index. If a character set is not defined for an individual index using the `indexvdkset` command, the value in this VDK setting is used. See "indexvdkset" on page 198. See the *Verity Locale Configuration Guide* for acceptable values.
`Memory Usage:`	
`Max Memory Size (KB)`	The maximum amount of memory in kilobytes that the K2 Server can use. This value is passed to the VDK sessions.
`Max Files`	The maximum number of file handles to hold open for each VDK session.
`Max Memory Percentage`	The maximum amount of memory, in terms of percentage, that the K2 Server can use. This value is passed to the VDK sessions.
`Number of 1K Pages`	The default number of pages (1024 bytes) used by each thread (VDK session) for disk caching of the collection partitions.

Managing K2 Ticket Servers

The commands in this section manage K2 Ticket Servers. A *K2 Ticket Server* is a K2 service that is used to implement index-level security. It is a centralized authentication service that stores information for users that have been authenticated to LDAP, Windows or UNIX, or secure repositories. When users authenticate to a security model, they receive a "ticket." The K2 Ticket Server monitors users and gives them access only to the indexes for which they have the correct ticket.

After adding a K2 Ticket Server to the system, you must:

- attach the K2 Ticket Server to an Administration Server, K2 Broker, or K2 Server. See "ticketattachset" on page 111.
- configure a login module and persistent store module for the K2 Ticket Server using the `ticketlmset` and `ticketpsmset` commands. See "ticketlmset" on page 170 and "ticketpsmset" on page 182.

For more information on K2 Ticket Servers, see the *Verity K2 Dashboard Administrator Guide* and the *Verity K2 Getting Started Guide*.

This section describes the following commands:

- ticketset
- ticketget

ticketset

The `ticketset` command enables you to add a K2 Ticket Server to an Administration Server, or modify information for an existing K2 Ticket Server.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for K2 Ticket Server, or add a new K2 Ticket Server. The following options are available: `0` = Update existing information for a K2 Ticket Server. `1` = Insert a new K2 Ticket Server.
`Ticket Alias`	The unique identifier for the K2 Ticket Server. The K2 Ticket Server alias should include the host name to ensure that each server has a unique name in a multi-host K2 domain. For example, an appropriate name would be `myhost_ticketserver1`.
`Port`	The port used by the K2 Ticket Server. This value can only be changed when the K2 Ticket Server is not attched to the Administration Server.
`Description`	A text description of the K2 Ticket Server.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Ticket Server is attached. In a multi-host system, this is any Administration Server alias. In a single-host system, this is the Master Administration Server alias for this machine. This is normally the machine's host name. This setting cannot be changed for an existing K2 Ticket Server.
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	Deprecated in K2 V5.0. This value is not used.

Argument	Description
`Number of Senders`	The number of senders used by the K2 Ticket Server. Senders send information from the K2 Ticket Server to the other K2 components. The default is 5.
`Number of Listeners`	The number of listener threads used by the K2 Ticket Server. This controls the number of concurrent connections to the server. The default is 200
`Ticket Timeout`	The length of time before a ticket expires. The default and minimum is 10 minutes.
`Ticket Timeout Check`	The interval (in minutes) after which to check for expired tickets. The default is 5 minutes.
`Use Login Credentials`	If enabled, the user's login credentials are used to attempt to authenticate to the repository. This must be enabled if pre-authentication mode is enabled.
`Enable Pre-Authentication? (y|n) (No):`	Enables pre-authentication mode in the K2 Ticket Server. If pre-authentication mode is enabled, K2 only requires a valid user name to access repositories. Authentication is not performed. For information on implementing single sign-on and pre-authentication with a Netegrity SiteMinder Identity and Access Management (IAM) system, see the technical note *Integrating K2 with Netegrity SiteMinder* available in the documentation library installed with K2.
`Client Method (0):`	If pre-authentication is enabled, you see this prompt. Reserved for future use. Press the `Enter` key.
`External Authentication Data:`	
`Name():`	If pre-authentication is enabled, you see this prompt. Reserved for future use. Press the `Enter` key.
`Value():`	If pre-authentication is enabled, you see this prompt. Reserved for future use. Press the `Enter` key.
`User Defined():`	If pre-authentication is enabled, you see this prompt. Reserved for future use. Press the `Enter` key.

Argument	Description
`Filter External USER_PARAMS keys from Login Credentials`	If this is enabled, custom authentication data passed into the login credentials through the `USER_PARAMS` key is removed from K2. By default, this is disabled, and the information is passed to K2. If pre-authentication mode is enabled, and the pre-authentication data is passed in through the `EXTENSION_KEY`, it is recommended you set this to `Y`. If the pre-authentication data is passed in through the `USER_PARAMS` key, set this to `N`.

ticketget

The `ticketget` command enables you to view information about a K2 Ticket Server.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the K2 Ticket Server.
`Port`	The port used by the K2 Ticket Server.
`ServerSpec`	The specification for the machine hosting the K2 Ticket Server.
`Description`	A text description of the K2 Ticket Server.
`Configured State`	The state of the K2 Ticket Server according to the `admin`*N*`.xml` file. `offline` = The K2 Ticket Server is offline and is not available. `online` = The K2 Ticket Server is online and is available. `hidden` = The K2 Ticket Server is hidden. It is not available to users, but can be operated on by administrators.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful if you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Ticket Server is attached.
`VDK Home`	Deprecated in K2 V5.0. This value is not used.
`CharMap`	Deprecated in K2 V5.0. This value is not used.
`Number of Senders`	The number of senders used by the K2 Ticket Server. Senders send information from the K2 Ticket Server to the other K2 components.

Argument	Description
`Number of Listeners`	The number of listener threads used by the K2 Ticket Server. This controls the number of concurrent connections to the server.
`Ticket Timeout`	The length of time before a ticket expires. The default is 10 minutes.
`Ticket Timeout Check`	The interval (in minutes) after which to check for expired tickets. The default is 5 minutes.
`Login Module Type`	The type of the login module. The following options are available: `Unknown` = The type is unknown. `Windows` = The login module uses Windows authentication. `Unix` = The login module uses UNIX authentication. `LDAP` = The login module uses LDAP authentication. `User Defined` = The login module is user defined.
`Persist Module Type`	The type of the PSM (persistent store module). The following options are available: `Unknown` = The PSM type is unknown. `Default PSM` = The PSM uses the default dynamic memory store. `LDAP` = The PSM uses LDAP. `User Defined` = The PSM is user defined.
`Use Login Credentials`	Indicates whether login credentials are used to attempt to authenticate to the repository.
`Filter External USER_PARAMS keys from Login Credentials`	If enabled, custom authentication data passed into the login credentials through the `USER_PARAMS` key is removed from K2. By default, this is set to `No`, and the information is passed to K2.

Argument	Description
`Enable Pre-Authentication`	Enables pre-authentication mode for the K2 Ticket Server. If pre-authentication mode is enabled, K2 only requires a valid user name to access repositories. Authentication is not performed.
`Client Method`	Reserved for future use.
`External Authentication Data:`	
`Name()`	Reserved for future use.
`Value()`	Reserved for future use.
`User Defined()`	Reserved for future use.

Managing K2 Index Servers

The commands in this section manage K2 Index Servers. A *K2 Index Server* is a K2 service that performs maintenance of parametric indexes, taxonomies, and topic sets.

A K2 Index Server can also be configured using the VAdministration Java API. See the Javadoc for VAdministration for more information.

For more information on K2 Index Servers, see the *Verity Organization Developer's Kit Programming Guide* and the *Verity K2 Dashboard Administrator Guide*.

This section describes the following commands:

- indexerset
- indexerget

Adding a K2 Index Server

To add a K2 Index Server to an Administration Server, follow these steps:

1. Start `rcadmin` from a command prompt.
2. If the Administration Server has a K2 Ticket Server attached to it, type `login` at the `rcadmin` prompt. Enter the user name, password and domain name (optional).
3. Type `indexerset` at the `rcadmin` prompt.
4. Type `1` to add a new K2 Index Server.
5. Enter the unique identifier for the K2 Index Server.

   The K2 Index Server alias should include the host name to ensure that each server has a unique name in a multi-host K2 domain. For example, an appropriate name would be `myhost_indexserver1`.
6. Enter the port number the K2 Index Server will use to listen for requests.

   This can be any available port number from 1023 to 32768. It is recommended you use values from 9960 to 9979.
7. Optionally, enter a description for this K2 Index Server.
8. Optionally, enter a bind IP address for machines which have multiple network cards.
9. Enter the unique identifier for the Administration Server to which the K2 Index Server will be attached.

In a multi-host system, this is any Administration Server alias. In a single-host system, this is the Master Administration Server alias for this machine. This is normally the machine's host name.

10. Enter the number of listeners. This controls the number of concurrent, connected clients to this instance of the K2 Index Server. The default is 35.

Note The Java library pools connections so the required number of connections from a single Java Virtual machine is constrained to the number of simultaneous requests sent to the K2 Index Server.

11. Enter the number of threads available for asynchronous calls. This controls the number of concurrent, processing-intensive requests, such as populating a parametric index or indexing a topic set into a collection. A general rule is to enter a value that is two times the number of CPUs you want used by this K2 Index Server. When more than the allotted number of requests are made at one time, the excess requests wait for an available slot. The default is 3.

12. Enter the desired number of synchronous threads. This controls the number of concurrently opened work spaces. It generally corresponds to concurrently active users. The default is 25.

13. Enter the access level required to use this instance of the K2 Index Server. K2 Index Server authentication is done using the K2 Ticket Server attached to the Administration Server. Valid values are:

    0 = Authenticated. Only authenticated users are allowed to access the K2 Index Server. This is the default.

    1 = Anonymous. Both authenticated and un-authenticated (anonymous) users are allowed to access the K2 Index Server. This setting must be used if there is no K2 Ticket Server attached to the Administration Server.

    2 = Administration. Only registered K2 administrators are allowed to access the K2 Index Server.

Note You can use the trusted clients commands to further restrict access to the K2 Index Server. The *trusted client* feature enables you to control which network addresses can be used to open communications with a K2 Index Server. By using the trusted client settings, you can restrict the IP addresses from which incoming connections will be accepted. See "Managing Trusted Clients" on page 164 for more information.

14. Type `y` to save your changes. Type `n` to cancel.

Example

```
C:\>rcadmin
K2Admin Server: server1:9950 connected. Type ? for help.
RCADMIN - Verity, Inc. Version 6.1
rcadmin>login
User:test
Password:*******
Windows NT Domain (optional):
User <test> logged in.
rcadmin> indexerset
Modify Type (Update=0, Insert=1):1
Index Server Alias:myhost_indexserver1
Port:9960
Description:The Index Server on server1
Bind:
Admin Alias:server1
Number of Listeners:
Number of Async. Threads:
Number of Sync. Threads:
Access Type(Auth=0,Anonymous=1,Admin=2):2
Save changes? [y|n]:y
<<Return>> SUCCESS
```

Starting a K2 Index Server

After adding the K2 Index Server to the Administration Server, you must start it manually, or automatically when the Administration Server runs.

- To start the K2 Index Server manually, type the following at a command prompt:

  `k2index -alias` *`index_server_alias`*

- To start the K2 Index Server automatically, perform a full restart using the `servicesignal` command. See "servicesignal" on page 119. A restart using this command also adds the service as a watched service. K2 automatically restarts a watched service if it finds it stopped.

Registering Indexes for Use with a K2 Index Server

When you introduce an index to the K2 system—a collection, knowledge tree, parametric index, or recommendation index—you must register the index to the Administration Server on the host. If the index will be used by a K2 Server, it must also be attached to the K2 Server using the `indexattach` command. Unlike a K2 Server, an index is not attached to a K2 Index Server before it can be used.

The following indexes are supported by the K2 Index Server:

- collection
- parametric index
- topic set

For more information on registering these indexes using `rcadmin`, see "Managing Collections" on page 205, "Managing Topic Sets" on page 224, and "Managing Parametric Indexes" on page 228.

indexerset

The `indexerset` command enables you to add a K2 Index Server to an Administration Server, or modify information for an existing K2 Index Server.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for a K2 Index Server, or add a new K2 Index Server. The following options are available: `0` = Update existing information for a K2 Index Server. `1` = Insert a new K2 Index Server. Inserting a K2 Index Server creates the following new directories: *`dataDir`*`/services/`*`IndexServiceAlias`*`/odk` *`dataDir`*`/services/`*`IndexServiceAlias`*`/log` where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).
`Index Server Alias`	The unique identifier for the K2 Index Server. The K2 Index Server alias should include the host name to ensure that each server has a unique name in a multi-host K2 domain.
`Port`	The port used by the K2 Index Server. It is recommended you use values from 9960 to 9979. This value cannot be changed for an existing K2 Index Server.
`Description`	A text description of the K2 Index Server.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful when you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server to which the K2 Index Server is attached. In a multi-host system, this is any Administration Server alias. In a single-host system, this is the Master Administration Server alias for this machine. This is normally the machine's host name.

Argument	Description
`Number of Listeners`	Controls how many clients and other K2 services can connect to this K2 service. This value is considered maximum because K2 manages the number of listeners—depending on the load—up to this value. The default is 35.
`Number of Asynch. Threads`	This controls the number of concurrent, processing-intensive requests, such as populating a parametric index or indexing a topic set into a collection. A general rule is to enter a value that is two times the number of CPUs you want used by the K2 Index Server. When more than the allotted number of requests are made at one time, the excess requests wait for an available slot. The default is 3.
`Number of Synch. Threads`	This controls the number of concurrently opened work spaces. It generally corresponds to concurrently active users. The default is 25.
`Access Type`	The user level required to access the K2 Index Server. Authentication is done using the K2 Ticket Server attached to the K2 Administration Server. `0` = Authenticated. Only authenticated users are allowed to access the K2 Index Server. `1` = Anonymous. Both authenticated and un-authenticated (anonymous) users are allowed to access the K2 Index Server. This setting must be used if there is no K2 Ticket Server attached to the K2 Administration Server. `2` = Administration. Only registered K2 administrators are allowed to access the K2 Index Server. The default is 1 (Anonymous).

indexerget

The `indexerget` command enables you to view information about K2 Index Servers.

Input

Enter the following information when prompted:

Argument	Description
`Index Server Alias`	The unique identifier for the K2 Index Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the K2 Index Server.
`Port`	The port used by the K2 Index Server.
`ServerSpec`	The specification for the machine hosting the K2 Index Server, for example, *`hostname`*`.verity.com:9960`.
`Description`	A text description of the K2 Index Server.
`Configured State`	The state of the K2 Index Server according to the `admin`*`N`*`.xml` file. `offline` = The K2 Index Server is offline and is not available. `online` = The K2 Index Server is online and is available.
`Internal Service:`	Indicates whether the service is reserved for internal K2 functionality. For example, an Index Server created for use by the Verity Business Console is an internal service.
`Bind`	If an IP address is specified here, then only requests directed to that IP address are accepted. Specifying an IP address here is useful when you have more than one network card and want to only accept requests on one of them.
`Admin Alias`	The unique identifier for the Administration Server for the K2 Index Server.
`Listeners`	Controls how many clients and other K2 services can connect to this K2 service. This value is considered maximum because K2 manages the number of listeners—depending on the load—up to this value. The default is 35.

Argument	Description
`Asynch. Threads`	This controls the number of concurrent, processing-intensive requests, such as populating a parametric index or indexing a topic set into a collection. When more than the allotted number of requests are made at one time, the excess requests wait for an available slot.
`Synch. Threads`	This controls the number of concurrently opened work spaces. It generally corresponds to concurrently active users.
`Access` Type	The user level required to access the K2 Index Server. Authentication is done using the K2 Ticket Server attached to the K2 Administration Server. `0` = Authenticated. Only authenticated users are allowed to access the K2 Index Server. `1` = Anonymous. Both authenticated and non-authenticated (anonymous) users are allowed to access the K2 Index Server. This setting must be used if there is no K2 Ticket Server attached to the K2 Administration Server. `2` = Administration. Only registered K2 administrators are allowed to access the K2 Index Server. The default is `0` (Authenticated).

Managing External K2 Services

This group of commands enables you to attach or detach external K2 services.

Note To configure external services, you must have a valid license for the Multi-domain feature.

Attaching external services enables the local K2 system to search K2 Brokers and K2 Servers in another *K2 domain*. K2 domains can be independent K2 systems in different geographic location or departments, or different versions of K2. For more information, see the *Verity K2 Installation and Setup Guide* and the *Verity K2 Migration Guide.*

When attaching services within a K2 domain, K2 prevents loops from being established. For example, if a K2 domain search hierarchy consists of

K2 Broker A -> K2 Broker B ->K2 Server C,

then K2 will not allow K2 Broker A to be attached below K2 Broker B. However, when attaching external services, K2 cannot prevent such loops. So it is your responsibility to avoid them.

When you attach a version 4.5 or higher external K2 service, K2 verifies that the alias exists, and retrieves the host and port of the external service from the Administration Server. If the external service is an earlier version, K2 informs you which INI entries need to be changed.

Note To name K2 domains, use the `hostdomainset` command. See "hostdomainset" on page 49.

This section describes the following commands:

- extbrokerattachset
- extticketattachset
- extsvcticketattachset
- extbrokerdetach
- extticketdetach
- extsvcticketdetach

extbrokerattachset

The `extbrokerattachset` command enables you to attach an external K2 service below a local K2 Broker. The external service is usually a K2 Broker, although it can also be a K2 Server.

To detach the external service, use the `extbrokerdetach`. See "extbrokerdetach" on page 99.

Note After attaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command. In K2 version 4.0.3, restart the external service manually from the external service's host machine.

Input

Enter the following information when prompted:

Argument	Description
`Local Broker Alias`	The unique identifier for the local K2 Broker to which the external service will attach.
`External Service Alias`	The unique identifier for the external service that will attach to the local K2 Broker.
`Version`	The version of the external service (for example, 5.5, or 6.1).
`Modify Type`	Indicates whether to modify existing configuration information for an external service, or create new configuration information for an external service. The following options are available: `0` = Update existing information for an external service. `1` = Insert a new external service.
`External Master Administration Server Information:`	This only appears if the external service is version 4.5 or higher.
`Host Name`	The name of the host used by the external Master Administration Server. This only appears if the external service is version 4.5 or higher.

Argument	Description
`Port`	The port number used by the external Master Administration Server. This only appears if the external service is version 4.5 or higher.
`Login to external K2 Domain:`	The prompts in this section only appears when the external service uses a K2 Ticket Server.
`User`	The user name used to log in to the external domain. This need not be the same user name that is used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods.
`Password`	The password for `User`.
`Windows NT Domain`	The Windows domain for `User`. This is optional.
`External Services Settings:`	
`K2 Domain Name`	The name of the K2 domain to which the external service belongs. If you are attaching a version 4.5 external service, a default is not provided. You must enter the domain name. Leave this blank if the domain is unnamed. If you are attaching a version 5.0 or higher external service, `rcadmin` shows the current domain name. (Read only.)
`Serverspec`	The *`host`*`:`*`port`* specification that the local K2 Broker uses to contact the external service. If the external service is V4.0 or earlier, a default is not provided; and you must enter the specification. Normally you can accept the default. But you may need to alter *`host`* if, for example, the local domain needs to use hard-coded IP numbers because its DNS server does not recognize the external service.
`Description`	An optional text description of the external service.
`Number of Collection Threads`	The number of threads used by collections on the external service. The default is 3.

Argument	Description
`Number of Tree Threads`	The number of threads used by knowledge trees on the external service. The default is 3. This prompt only appears if the K2 service you are attaching is version 4.0 or higher.
`Number of Parametric Threads`	The number of threads used by parametric indexes on the external service. The default is 3. This prompt only appears if the K2 service you are attaching is version 5.0 or higher.
`Number of Profiler Threads`	The number of threads used by Profiling Engine on the external services. The default is 3. This prompt only appears if the K2 service you are attaching is version 5.0 or higher.
`Number of Recommendation Threads`	The number of threads used by Recommendation Engine on the external services. The default is 3. This prompt only appears if the K2 service you are attaching is version 5.0 or higher.
`Connection Timeout (msecs)`	The length of time to wait, in milliseconds, for a response before functions time out on the external service.
`External Service Connection Settings:`	
`Service Alias`	The alias used by the external service to identify the local K2 Broker. (Read only.)
`Serverspec`	The *`host`*`:`*`port`* specification that the external service uses to contact the local K2 Broker. Normally you can accept the default. But you may need to alter *`host`* if, for example, the external service needs to use hard-coded IP numbers because its DNS server does not recognize the K2 Broker.

If the external service is version 4.5.x, or higher, the service is attached. If the external service is version 2.2 or 4.0, then `rcadmin` shows the modifications you must make to the external service's INI file.

extticketattachset

The `extticketattachset` command enables you to attach an external K2 Ticket Server to a local K2 Broker or K2 Server.

The external K2 Ticket Server is used for gateway authentication (document-level security), not for logging in users (index-level security). The external K2 Ticket Server can use a different authentication method than the local K2 Ticket Server. For example, one can use Window authentication and the other can use LDAP authentication.

Login authentication (for index-level security) is always handled through the local K2 Ticket Server attached to the top level K2 Broker.

If an external K2 domain uses gateway authentication (for document-level security) then you must attach that domain's K2 Ticket Server as an external service.

For more information, see the *Verity K2 Migration Guide*.

To detach the K2 Ticket Server, use the `extticketdetach`. See "extticketdetach" on page 101.

Note After attaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the local K2 service to which the external K2 Ticket Server will attach.
`External Ticket Alias`	The unique identifier for the external K2 Ticket Server that will attach to the K2 service.
`External Ticket Version`	The version of the external K2 Ticket Server (for example, 5.5 or 6.1). Version 4.0.x K2 Ticket Servers are not supported as external services.

Argument	Description
`Modify Type`	Indicates whether to modify existing configuration information for an external K2 Ticket Server, or create new configuration information for an external K2 Ticket Server. The following options are available: `0` = Update existing information for an external K2 Ticket Server. `1` = Insert an new external K2 Ticket Server.
`External Master Administration Server Information:`	
`Host Name`	The name of the host used by the external Master Administration Server.
`Port`	The port number used by the external Master Administration Server.
`Login to external K2 Domain:`	This only appears if the external service uses a K2 Ticket Server.
`User`	The user name to use to log in to the external domain. This need not be the same user name that is used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods. This only appears if the external service uses a K2 Ticket Server.
`Password`	The password for `User`. This only appears if the external service uses a K2 Ticket Server.
`Windows NT Domain`	The Windows domain for `User`. This is optional. This only appears if the external service uses a K2 Ticket Server.

Argument	Description
`External Ticket Settings:`	
`Ticket Domain`	The name of the K2 domain that the external K2 Ticket Server is in. If you are attaching a version 4.5 external service, you must enter the domain name. `rcadmin` is unable to determine the current name (if any). Leave this blank if the domain is unnamed. If you are attaching a version 5.0 or higher external service, `rcadmin` shows the current domain name. (Read only.)
`Ticket Serverspec`	The *`host`*`:`*`port`* specification that the external K2 Ticket Server uses to contact the local service. Normally you can accept the default. But you may need to alter *`host`* if, for example, the external service needs to use hard-coded IP numbers because its DNS server does not recognize the external service.
`Ticket Description`	An optional text description of the external K2 Ticket Server.
`Number of Thread`	The number of threads used by the K2 service to connect to the external K2 Ticket Server.
`Connection Timeout (msecs)`	The length of time to wait, in milliseconds, for a response before functions time out on the external K2 Ticket Server.
`Service Settings:`	
`Service Alias`	The alias used by the external K2 Ticket Server to identify the local K2 service. (Read only.)
`Service Serverspec`	The *`host`*`:`*`port`* specification that the external K2 Ticket Server uses to contact the local K2 service.

extsvcticketattachset

The `extsvcticketattachset` command enables you to attach an external K2 service to a local K2 Ticket Server.

To detach the K2 service, use the `extsvcticketdetach`. See "extsvcticketdetach" on page 102.

Note After attaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the local K2 Ticket Server to which the external service will attach.
`External Service Alias`	The unique identifier for the external K2 service that will attach to the local K2 Ticket Server.
`External Service Version`	The version of the external service (for example, 5.5 or 6.1).
`External Master Administration Server Information:`	
`Host Name`	The name of the host used by the external Master Administration Server.
`Port`	The port number used by the external Master Administration Server.
`Login to external K2 Domain:`	This only appears if the external service uses a K2 Ticket Server.
`User`	The user name to log in to the external domain. This need not be the same user name as used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods. This only appears if the external service uses a K2 Ticket Server.

Argument	Description
`Password`	The password for `User`. This only appears if the external service uses a K2 Ticket Server.
`Windows NT Domain`	The Windows domain for `User`. This is optional. This only appears if the external service uses a K2 Ticket Server.
`Modify Type`	Indicates whether to modify existing configuration information for an external service, or create new configuration information for an external service. The following options are available: `0` = Update existing information for an external service. `1` = Insert a new external service.
`Ticket Domain`	The name of the K2 domain that the local K2 Ticket Server is in. (Read only.)
`Ticket Serverspec`	The *`host`*`:`*`port`* specification that the external K2 service uses to contact the local K2 Ticket Server. Normally you can accept the default. But you may need to alter *`host`* if, for example, the external K2 service needs to use hard-coded IP numbers because its DNS server does not recognize the local K2 Ticket Server.
`Ticket Description`	An optional text description of the local K2 Ticket Server.
`Number of Threads`	The number of threads used by the external K2 service to connect to the local K2 Ticket Server.
`Connection Timeout (msecs)`	The length of time to wait, in milliseconds, for a response before functions time out on the external K2 service.
`Service Settings:`	
`Service Alias`	The alias used by the external K2 service to identify the local K2 Ticket Server. (Read only.)
`Service Serverspec`	The *`host`*`:`*`port`* specification that the external K2 service uses to contact the local K2 Ticket Server.

extbrokerdetach

The `extbrokerdetach` command enables you to detach an external K2 service from a local K2 Broker.

Normally, when you detach an external service, K2 needs to contact the external service to update its configuration. If K2 is unable to contact it (because for example the network is down), the entries about the external service will be removed from the local K2 domain, but the external K2 domain will still have entries referring to the local K2 domain. To clean up those entries, you can attach the external service (when the external service is available) and then run the detach command again.

Note After detaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command. In K2 version 4.0.3, restart the external service manually from the external service's host machine

Input

Enter the following information when prompted:

Argument	Description
`Broker Alias`	The unique identifier for the local K2 Broker.
`External Alias to Detach`	The unique identifier for the external K2 service.
`External Domain`	The name of the external K2 domain. Leave this blank if the domain is unnamed.
`External Master Administration Server Information:`	
`Host Name`	The name of the host used by the external Master Administration Server. (You must use the Master Administration Server, not an Administration Server.)
`Port`	The port number used by the external Master Administration Server.

Argument	Description
`Login to external K2 Domain:`	The prompts in this section only appear if the external service uses a K2 Ticket Server.
`User`	The user name to log in to the external domain. This need not be the same user name as used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods.
`Password`	The password for `User`.
`Windows NT Domain`	The Windows domain for `User`. This is optional.

If the external service is version 4.5 or higher, the service are detached. If the external service is version 2.2 or 4.0, then `rcadmin` shows the modifications you must make to the external service's INI file.

extticketdetach

The `extticketdetach` command enables you to detach an external K2 Ticket Server from a local K2 service.

Normally, when you detach an external service, K2 needs to contact the external service to update its configuration. If K2 is unable to contact it (because for example the network is down), the entries about the external service will be removed from the local K2 domain, but the external K2 domain will still have entries referring to the local K2 domain. To clean up those entries, you can attach the external service (when the external service is available) and then run the detach command again.

Note After detaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the local K2 service.
`External Ticket Alias To Detach`	The unique identifier for the external K2 Ticket Server.
`External Ticket Domain`	The name of the external K2 Ticket Server domain. Leave this blank if the domain is unnamed.
`External Master Administration Server Information:`	
`Host Name`	The name of the host used by the external Master Administration Server. (You must use the Master Administration Server, not an Administration Server.)
`Port`	The port number used by the external Master Administration Server.

Argument	Description
`Login to external K2 Domain:`	
`User`	The user name to log in to the external domain. This need not be the same user name as used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods.
`Password`	The password for `User`.
`Windows NT Domain`	The Windows domain for `User`. This is optional.

extsvcticketdetach

The `extsvcticketdetach` command enables you to detach an external K2 service from a local K2 Ticket Server.

Normally, when you detach an external service, K2 needs to contact the external service to update its configuration. If K2 is unable to contact it (because for example the network is down), the entries about the external service will be removed from the local K2 domain, but the external K2 domain will still have entries referring to the local K2 domain. To clean up those entries, you can attach the external service (when the external service is available) and then run the detach command again.

Note After detaching an external service, restart the local and external service. In K2 version 4.5.1 and higher, perform a quick restart on both services using the `servicesignal` command.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the local K2 Ticket Server.
`External Service Alias To Detach`	The unique identifier for the external K2 service.
`External Ticket Domain`	The name of the external K2 service domain. Leave this blank if the domain is unnamed.

Argument	Description
`External Master Administration Server Information:`	
`Host`	The name of the host used by the external Master Administration Server. (You must use the Master Administration Server, not an Administration Server.)
`Port`	The port number used by the external Master Administration Server.
`Login to external K2 Domain:`	
`User`	The user name to log in to the external domain. This need not be the same user name as used to log in to the local domain. Furthermore, the two domains need not use the same authentication methods.
`Password`	The password for `User`.
`Windows NT Domain`	The Windows domain for `User`. This is optional.

Attaching and Detaching Services

The commands in this section attach or detach K2 services to/from other K2 services. When you add a K2 service, you must attach it to a K2 Broker or K2 Server.

This section describes the following commands:

- servicedetach
- brokerattachset
- brokerattachlistget
- brokerattachget
- ticketattachset
- ticketattachlistget
- ticketattachget

servicedetach

The `servicedetach` command enables you to detach a K2 service from another K2 service.

Before using the `servicedel` command to delete a K2 service, you must use this command to detach all services above and below the K2 service in the K2 hierarchy.

Note If you detach a service, use the `servicesignal` command to perform a quick restart on the K2 Ticket Server (if applicable) and the service from which you are detaching. This applies the change to the system. See "servicesignal" on page 119.

Argument	Description
`Service Alias`	The unique identifier for the service from which the other service will be detached.
`Service Type For` *`alias`*	The type of service from which you are detaching: `a` = Administration Server. `b` = K2 Broker. `s` = K2 Server.
`Service Alias To Detach`	The unique identifier for the service to be detached. For a K2 Broker, the service alias can be a K2 Ticket Server or K2 Server. For a K2 Server, the service alias can be a K2 Ticket Server. For a K2 Administration Server, the service alias can be a K2 Ticket Server.

Example

For example, say you want to delete a K2 Server named “S1” that is part of a K2 system. S1 is currently attached below a K2 Broker named “B1,” and above a K2 Ticket Server named “T1.” To remove S1 from the system:

1. Detach the broker “B1:”

```
rcadmin> servicedetach
Service Alias:b1
Service Type For "b1" [(a)dmin|(b)roker|(s)erver]:b
Service Alias To Detach:s1
Save changes? [y|n]:y
<<Return>> SUCCESS
```

2. Detach the ticket “T1:”

```
rcadmin> servicedetach
Service Alias:s1
Service Type For "s1" [(a)dmin|(b)roker|(s)erver]:s
Service Alias To Detach:t1
Save changes? [y|n]:y
<<Return>> SUCCESS
```

3. Perform a quick restart on the K2 Server “S1:”

```
rcadmin> servicesignal
Service Alias:s1
Type of signal (Quick=0,Full=1,Shutdown=2,Kill=3):0
Save changes? [y|n]:y
<<Return>> SUCCESS
```

4. Delete the K2 Server “S1:”

```
rcadmin> servicedel
Service Alias:s1
Service Type For "s1"
[(a)dmin|(b)roker|(s)erver|(t)icket|(i)ndexServer]:s
Save changes? [y|n]:y
<<Return>> SUCCESS
```

brokerattachset

The `brokerattachset` command enables you to attach a local K2 Server or K2 Broker below a specified K2 Broker.

To attach an external K2 Broker, use the `extbrokerattachset` command. See "extbrokerattachset" on page 91.

Note Circular references are not allowed in the K2 hierarchy. For example, if Broker B has Server S attached below it, then Broker B cannot be attached below Server S. Use `brokerattachlistget` command to determine the allowable attachments. See "brokerattachlistget" on page 109.

Input

Enter the following information when prompted:

Argument	Description
`Broker Alias`	The unique identifier for the K2 Broker to which the K2 service will attach.
`Modify Type`	Indicates whether to modify existing information for a service, or insert information for a new service. The following options are available: `0` = Update existing information for a service. `1` = Insert information for a new service.
`Service Alias`	The unique identifier for the K2 service that will attach to the K2 Broker.
`Service Type For alias`	The type of service being attached. The following options are available: `b` = K2 Broker. `s` = K2 Server.
`Number of Collection Threads`	The number of threads to be used by collections. The default is 3.
`Number of Tree Threads`	The number of threads to be used by knowledge trees. The default is 3.
`Number of Parametric Threads`	The number of threads to be used by parametric indexes. The default is 3.

Argument	Description
`Number of Profiler Threads`	The number of threads to be used by Profile Nets. The default is 2.
`Number of Recommendation Threads`	The number of threads to be used by the Recommendation Engine. The default is 3.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.

brokerattachlistget

The `brokerattachlistget` command enables you to view a list of services that can be attached below a specified K2 Broker.

Circular references are not allowed in the K2 hierarchy. For example, if K2 Broker A has K2 Broker B attached below it, then K2 Broker A cannot be attached below K2 Broker B. This command allows you to determine the allowable attachments.

Input

Enter the following information when prompted:

Argument	Description
`Broker Alias (press Enter for all brokers)`	The unique identifier for the K2 Broker whose information you want to display. To display information about all K2 Brokers, press Enter.
`Type of Service to Get`	The type of services to display. The following options are available: `b` = display all K2 Brokers that can be attached. `s` = display all K2 Servers that can be attached.

Output

If you specified a `Broker Alias`, `rcadmin` shows the following information:

Argument	Description
`Available services`	The number of services that can be attached.
`Service Alias`	The aliases of the services that can be attached to the K2 Broker.

brokerattachget

The `brokerattachget` command enables you to view information about K2 Brokers or K2 Servers attached below a specified K2 Broker.

Input

Enter the following information when prompted:

Argument	Description
`Broker Alias`	The unique identifier for the K2 Broker.
`Service Alias (press Enter for all services attached)`	The unique identifier for the service whose information you want to display. To display information about all the services attached to the Broker, press Enter.

Output

If there are services attached, `rcadmin` shows the following information:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type`	The type of service. The following options are available: `Broker` = a K2 Broker. `Server` = a K2 Server.
`External Service`	`1` indicates that this is an external service (the service is not in the local K2 domain). `0` indicates that this service is in the local K2 domain.
`Service Domain`	The name of the K2 domain that the service is in.
`Service ServerSpec`	The *`host`*`:`*`port`* specification that the external service uses to contact the local K2 Broker.
`Service Description`	A text description of the service.
`Service Version`	The version of the external service.
`Number of Collection Threads`	The number of threads used by collections.
`Number of Tree Threads`	The number of threads used by knowledge trees.

Argument	Description
`Number of Parametric Threads`	The number of threads used by parametric indexes.
`Number of Profiler Threads`	The number of threads used by Profile Nets.
`Number of Recommendation Threads`	The number of threads to be used by the Recommendation Engine.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.

ticketattachset

The `ticketattachset` command enables you to attach a local K2 Ticket Server to a K2 service.

To attach an external K2 Ticket Server, use the `extticketattachset` command. See "extticketattachset" on page 94.

Note If you attach a K2 Ticket Server, use the `servicesignal` command to perform a quick restart on the K2 Ticket Server and on the service to which you are attaching. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.
`Modify Type`	Indicates whether to modify existing information for a K2 service, or insert information for a new K2 service. The following options are available: `0` = Update existing information for a K2 service. `1` = Insert information for a new K2 service.
`Service Alias`	The unique identifier for the service to which the K2 Ticket Server will be attached.

Argument	Description
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server.
`Number of Threads`	The number of threads used by the K2 service to connect to the K2 Ticket Server.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.

ticketattachlistget

The `ticketattachlistget` command enables you to view a list of K2 Ticket Servers that can be attached to a K2 service.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service. `rcadmin` shows information about all the K2 Ticket Servers that can be attached below this service. To display all the K2 Ticket Servers that can be attached below any service, press Enter.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = Administration Server. `b` = K2 Broker. `s` = K2 Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Available Ticket Servers`	The number of K2 Ticket Servers. This is 0 if there are no K2 Ticket Servers that are attached.
`Alias`	The unique identifier for each K2 Ticket Server.

ticketattachget

The `ticketattachget` command enables you to view information about K2 Ticket Servers that are attached to a K2 service.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service. `rcadmin` shows information about K2 Ticket Servers that are attached to this service.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server.
`Ticket Server Alias (or press Enter for all ticket Services)`	The unique identifier for the K2 Ticket Server for which you want to display information. To display information about all the K2 Ticket Servers attached to this service, press Enter.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for this K2 Ticket Server.
`Number of Threads`	The number of threads used by the K2 service to connect with the K2 Ticket Server.
`Connection Timeout`	The length of time to wait, in milliseconds for a response before functions time out.
`External Ticket`	`1` indicates that this K2 Ticket Server is an external service (in other words, that it is not in the local K2 domain). `0` indicates it is in the local K2 domain.
`Domain`	The name of the K2 domain that the K2 Ticket Server is in.
`ServerSpec`	The *`host`*`:`*`port`* specification of the K2 Ticket Server.
`Description`	A text description of the K2 Ticket Server.
`Version`	The version of the K2 Ticket Server.
`Number of Threads`	The number of threads used by the K2 Ticket Server.
`Connection Timeout`	The length of time to wait, in milliseconds, for a response before functions time out.

Starting and Stopping K2 Services

The commands in this section enable you to stop and start K2 services using the Verity Control Module (VCM). They also enable you to have K2 automatically stop or start services by setting watched services.

This section describes the following commands:

- adminlaunch
- adminsignal
- adminkilllaunched
- servicesignal
- servicestartup
- servicewatchedstate
- servicegetpid
- servicestateget
- wsadd
- wsdel
- wsdelall
- wsget

adminlaunch

The `adminlaunch` command enables you to launch K2 commands using `rcadmin`. See also the "adminkilllaunched" on page 118.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this service.
`Command`	The name of the command to run. For example, `mkvdk`. The commands that can be launched are those listed in the `[Launch]` section of the `verity.cfg` file. For more information on `verity.cfg`, see the *Verity K2 Dashboard Administrator Guide*.

`Parameters`	The arguments to pass with the command. This is optional.
	For example, typing `mkvdk` at the `Command` prompt, and typing
	`-create -collection my_collection`
	at the `Arguments` prompt is the same as typing the command and arguments on the command-line:
	`mkvdk -create -collection my_collection`
	This creates a new, empty collection.

`Launch as desktop application (Windows only)? (y|n) (No):`	Indicates whether the command is launched as a desktop application, or is run as a "hidden" process. If you enter `y`, the specified process runs in a window on the desktop and appears in the process list. If you enter `n`, the process only appears in the process list. This prompt applies to Windows only.

Output

`rcadmin` shows the following information:

Argument	Description
`PID`	The process ID of the command.

adminsignal

The `adminsignal` command enables you to send a "shutdown," "restart all services," or "refresh watched service" command to an Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`Type of signal`	The type of signal to send to the Administration Server. The following options are available: `2` = Shutdown. Shuts down the Administration Server. You must exit `rcadmin` after executing this command. `3` = Watched Service Refresh. Tells the watched services to refresh their watched service state from the `admin`*N*`.xml` file. `4` = Restart all servers. Restarts the K2 services controlled by the specified Administration Server, and adds them as watched services. K2 automatically restarts a watched service if it finds it stopped. If a K2 Service is set to run manually (using the `servicewatchedstate` command), and is then started using the `adminsignal` command, the state is changed to `AutoRun`. If you choose this option, and are using a K2 Ticket Server, you must re-start `rcadmin` and log back in again. This is because your ticket will expire when the K2 Ticket Server is stopped and restarted.

adminkilllaunched

The `adminkilllaunched` command enables you to kill a process that was launched using the `adminlaunch` command.

Note If you kill a service that is monitored by the Administration Server (a *watched service*), the service is automatically restarted by the Administration Server. To properly stop a K2 Service, use the `servicesignal shutdown` command.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this process.
`Process ID`	The ID of the process to kill.

servicesignal

The `adminsignal` command enables you to send a "restart," "kill," or "shutdown" command to a K2 service. When you restart a K2 service using this command, the service is added to the watched services list. An Administration Server monitors watched services, and starts and stops them as necessary.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 service. Type `@Controller` to start or stop the K2 Spider Controller.
`Type of signal`	The type of signal to send. The following options are available: `0` = Do a quick restart. A quick restart resets the K2 Server with minimal impact to its service. During a quick restart, the K2 Server re-reads the settings in the local K2 Administration and updates its configuration. The K2 Server does not have to stop. Client connections are locked and clients may experience a brief service delay. A quick restart also adds the service as a watched service. K2 automatically restarts a watched service if it finds it stopped. `1` = Do a full restart. A full restart shuts down the K2 Server and then restarts it so that a changed parameter can take effect. You will experience a brief outage while the K2 Server shuts down and restarts. A full restart also adds the service as a watched service. K2 automatically restarts a watched service if it finds it stopped. If the service is set to run manually (using the `servicewatchedstate` command), and is then started using the `servicesignal` command (option 1 "full restart"), the state is changed to `AutoRun`. `2` = Shut down. Tells the service to shut down. If the service is a watched service, it may take up to the entire polling interval to shut down. `3` = Kill. Immediately ends the service.

Example

```
rcadmin> servicesignal
Service Alias:myhost_indexerserver1
Type of signal (QuickRestart=0,FullRestart=1,Shutdown=2,Kill=3):1
Save changes? [y|n]:y
<<Return>> SUCCESS
```

servicestartup

The `servicestartup` command enables you to start a K2 service.

Note The `servicestartup` command restarts the specified service, and adds it as a watched service. K2 automatically restarts a watched service if it finds it stopped. If a service is set to run manually (using the `servicewatchedstate` command), and is then started using the `servicestartup` command, K2 starts the service, but the service remains in manual mode.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service to start. You can determine the aliases of services by using the `hostview` or `hierarchyview` command. See "hostview" on page 39 and "hierarchyview" on page 38.
`Parameter Override`	The arguments that override the default start-up arguments. This is optional. By default, K2 uses `-alias` *`Service_Alias`* when starting a K2 service.

servicewatchedstate

The `servicewatchedstate` command enables you to add a service to the watched service list, so that it can be started, stopped, and monitored by the K2 administration process. It also enables you to specify how K2 should respond when a service is running or stopped.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service.
`Desired State`	The desired state. The following options are available: `0` = AutoStop. K2 automatically stops the service if it finds it running. `1` = AutoRun. K2 automatically starts the service if it finds it stopped. `2` = Manual. K2 does not watch the service.

servicegetpid

The `servicegetpid` command enables you to view the process ID of a K2 service.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service.

Output

`rcadmin` shows the following information:

Argument	Description
`PID`	The process ID of the service.

wsadd

The `wsadd` command enables you to add a K2 service to the *watched service* list for an Administration Server. An Administration Server monitors watched services, and starts and stops them as necessary.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the service is attached.
`Service Alias`	The unique identifier for the service to be added to the watched service list. For example, `broker1`. Type `@Controller` to watch the K2 Spider Controller.
`Polling Interval`	The interval, in seconds, between K2's polls of this service to see if its state is correct. The default is 120 seconds.
`Desired State`	The desired state of the watched service. The following options are available: `0` = AutoStop. K2 automatically stops the service if it finds it running. `1` = AutoRun. K2 automatically starts the service if it finds it stopped. `2` = Manual. K2 does not watch the service.
`Path`	The directory path to the executable to be used if K2 needs to start the service. For example, on Windows it might be: `C:\program files\verity\k2\_nti40\bin\k2server.exe`
`Command Parameter`	Parameters to be added to executable on startup. For example: `-alias server1` (for a K2 Server) `-alias broker1` (for a K2 Broker) `-alias ticket1` (for a K2 Ticket Server) `-port 9800 -Controller` (for a K2 Spider)

wsdel

The `wsdel` command enables you to delete a service from an Administration Server's watched service list.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the watched service is attached.
`Service Alias`	The unique identifier for the watched service you want to delete.

wsdelall

The `wsdelall` command enables you to delete all services from an Administration Server's watched services list.

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.

wsget

The `wsget` command enables you to view information about watched services on an Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the watched service is attached.
`Service Alias`	The unique identifier for the watched service whose status information you want to view.

Output

`rcadmin` shows the following information:

Argument	Description
`Service Alias`	The unique identifier for the watched service.
`Polling Interval`	The interval, in seconds, between K2's polls of this service to see if its state is correct. The default is 120 seconds.
`Server State`	The desired state of the watched service. The following options are available: `AutoStop` = K2 sets the watched service to the stopped state. `AutoRun` = K2 sets the watched service to the running state. `Manual` = K2 does not watch the service. For example, if you delete a watched service using `wsdel`, and then use the `wsget` command, `rcadmin` returns the state of `manual`.
`Path`	The directory path to the executable to be used if K2 needs to start the service. For example, on Windows it might be: `C:\program files\verity\k2\_nti40\bin\k2server.exe`
`Command Parameter`	Parameters to be added to executable on startup. For example: `-alias server1` (for a K2 Server) `-alias broker1` (for a K2 Broker) `-alias ticket1` (for a K2 Ticket Server) `-port 9800 -Controller` (for a K2 Spider)

Checking the State of a K2 Service

The command in this section check the configured and running state of a K2 service.

This section describes the following commands:

- servicecheckstate
- servicestateget

servicecheckstate

The `servicecheckstate` command enables you to check the `adminN.xml` to determine whether a K2 service is listed as online, offline, or unavailable.

Services that can be online/offline/unavailable are: Administration Servers, K2 Brokers, K2 Servers, K2 Ticket Servers and K2 Index Servers.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Service State`	The configured state of the service. The following options are available: `offline` = The service is offline and is not available. `online` = The service is online and is available. `hidden` = The service is hidden. It is not available to users, but can be operated on by administrators.

servicestateget

The `servicestateget` command enables you to ping a K2 service and view its running state.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service. Type `@Controller` to ping the K2 Spider Controller.

Output

`rcadmin` shows the following information:

Argument	Description

`State`	The running state of the service. The following options are available:
	`Host not available`
	`Starting`
	`Stopping`
	`Running`
	`Stopped`

Removing K2 Services

The command in this section removes configuration information for a K2 service.

This section describes the following commands:

- servicedel

servicedel

The `servicedel` command enables you to delete configuration information for K2 services. This only removes the entries in the `admin`*N*`.xml` file. It does not delete any files.

Before deleting a K2 service, you must stop the service and detach all K2 services below and above it in the K2 hierarchy. To stop a K2 service, use the `servicesignal` command (see "servicesignal" on page 119). To detach a K2 service, use the `servicedetach` command (see "servicedetach" on page 105).

Before deleting this service	Detach it from these services:
K2 Server	- K2 Brokers above it - K2 Ticket Servers below it
K2 Broker	- K2 Brokers above it - K2 Brokers below it - K2 Servers below it - K2 Ticket Servers below it
K2 Ticket Server	- K2 Brokers above it - K2 Servers above it - Administration Servers above it
Administration Server (master or non-master)	- K2 Brokers below it - K2 Servers below it - K2 Ticket Servers below it NOTE: You do not normally delete an Administration Server

Note A K2 Index Server is not attached to a K2 Server or K2 Broker.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service you want to delete.
`Service Type For` *`alias`*	The type of service you are deleting: `a` = Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.

Managing Recommendation Transaction Logs

The commands in this section manage the recommendation transaction log for a K2 Server or K2 Broker. This is a log of the transactions between the K2 Broker and its K2 Servers. A transaction is a modification of one or more entities in a recommendation index. For example, a transaction may make a document more relevant to a particular query due to user input.

This section describes the following commands:

- retxlogset

retxlogset

The `retxlogset` command enables you to configure the Recommendation Engine's transaction log. See the *Verity K2 Recommendation Engine Guide* for more information on the Recommendation Engine.

Note If you change any of these settings, perform a quick restart on the Administration Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Broker or K2 Server.
`Service Type For` *`alias`*	The type of service: `b` = K2 Broker. `s` = K2 Server.

Argument	Description
`RE Log Enable?`	Indicates whether K2 Broker/Server transactions are logged for document recommendation. If transaction logging is enabled, the K2 Server can receive requests from a VTransaction API call and log the information in an XML file. The default path of the XML file is *`dataDir`*`/host/serveralias/re` where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). The file name is `tr.xml`.
`RE Transaction Log Path`	The directory path to the transaction log.
`RE Transaction Log Max Per Hour`	The rate of transaction logging, that is, the rate at which the K2 Server or K2 Broker samples VTransaction requests. The sampling of requests is a useful feature in cases when the K2 Server or K2 Broker can receive a lot of requests, and it suffices to select and log only a subset of these requests. By default, the K2 Server or K2 Broker record every request. If this parameter is set to a value, such as 60, then the K2 Server or K2 Broker sample incoming requests at the rate of 60/hour, or 1/minute. If, in the same minute, 2 requests arrive at the K2 Server or K2 Broker, then the second request is skipped (or ignored). To avoid this situation, set the rate to a high value, such as 10000, or do not set it at all.

Configuring the Search Results Cache

The command in this section configures a K2 Broker or K2 Server so that it caches search results. This can improve the performance of your K2 system. When you enable search results caching, you can limit how many results are cached, how large the cache can grow, and when the cache times out.

This section describes the following commands:

- searchcacheset

searchcacheset

The `searchcacheset` command enables you to configure the search cache (also known as the results cache) for a K2 Broker or K2 Server.

Note If you change any values with this command, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.
`Search Cache Enable?`	Indicates whether this service caches search results. Caching results can improve the performance of your K2 system.

Argument	Description
`Search Cache Timeout`	The length of time to wait, in milliseconds, for a response before the search cache times out. The default is 600,000 milliseconds. In this case, a timeout occurs after 600 seconds (10 minutes), or when the cache overflows. The minimum value is 30,000 milliseconds (30 seconds).
`Max Memory Size Used by Search (bytes)`	The maximum size, in bytes, of the cache. The default value is 256MB. If you enter a value of less than 1.5 MB, K2 uses a value of 1.5 MB.

4

Managing Profile Services

This commands in this chapter manage K2 *Profile Service*s and *Profile Net*s.

- Introduction
- Configuring a Profile Service
- Configuring a Profile Net

Introduction

Organizations use Verity Profiler to build applications that classify documents. *Classification* is the process of assigning documents to categories in a taxonomy.

Verity Profiler is based on sophisticated technology that matches the terms and concepts in incoming documents to specific sets of stored queries. These query sets are interest profiles that can determine whether a given document is about a given subject. The documents can then be automatically assigned to the appropriate category or categories, trigger messaging systems, or be sent directly to the user who needs them.

A *Profile Service* runs on a K2 Server. Multiple Profile Services can attach to one K2 Server. However, unlike Verity collections, a Profile Service can only attach to one K2 Server. Profile Services store sets of queries, called *Profile Net*s. Each Profile Service must have at least one Profile Net.

In some cases, you may want to configure several Profile Nets for the same Profile Service. This is useful when you want to separate query sets. For example, you may want to track new documents for information on a competitor. For example, the Profile Service named *Competitor Profile* could store Profile Nets called *Products*, and another called *Financial Results*. The first Profile Net would contain queries such as your competitor's name and the names of its products. Likewise, the second Profile Net would contain queries for your competitor's name and terms such as "quarterly performance" and "revenues". If a document scores highly against the first query set, it likely shows information on your competitor's products. If another document scores highly against the second query set, it gives published information on your competitor's financial results.

`rcadmin` enables you to create Profile Services on a K2 Server, add Profile Net directories, and populate them with the necessary files. However, to create the necessary files, you must run the `mkprf` command line tool. For more information, see the *Verity Profiler Programming Guide*. You can use `rcadmin` to configure Profile Services and Nets, but only if you have a license for Verity Profiler.

To configure a Profile Service:

1. Add the Profile Service to a K2 Server using the `profileset` command. See "profileset" on page 135.
2. Add a Profile Net to the Profile Service using the `profilenetset` command. See "profilenetset" on page 141.
3. If necessary, you can change the state of the Profile Net after it is added using the `profilenetstateset`. See "profilenetstateset" on page 145.

Configuring a Profile Service

The commands in this section enable you to configure a Profile Service for a K2 Server.

This section describes the following commands:

- profileset
- profileget
- profiledel

profileset

The `profileset` command enables you to configure a Profile Service for a K2 Server.

Note If you add or change a Profile Service, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server to which the Profile Service is added.
`Profile Alias`	A unique identifier for the Profile Service. K2 supports multi-byte Profile Service names.
`Modify Type`	Indicates whether to modify an existing Profile Service, or insert a new one. The following options are available: `0` = Update an existing Profile Service. `1` = Insert a new Profile Service.
`Threads`	The number of threads used by this Profile Service. The default is 2.

Argument	Description
`Topic Set Path [Deprecated. Use Topic Set Alias]`	Press the `Enter` key at this prompt. The `Topic Set Path` prompt was deprecated in K2 V6.0. The topic set associated with a Profile Service is defined by the topic set alias.
`Topic Set Alias`	The unique identifier for the topic set used by this Profile Service. This is optional.
`Query Parser`	The default query parser used by the Profile Service. This default is used if another query parser is not specified when queries are added. The following options are available: `Simple` = Use the standard Verity syntax. `BoolPlus` = Use the explicit Verity syntax. `FreeText` = Use the Verity freetext query parser, which interprets free text using natural language processing tools. `Old Simple` = Use the old Verity simple query parser. This value is included for backward compatibility. `Old FreeText` = Use the old Verity freetext query parser. This value is included for backward compatibility. The default is the simple query parser.
`Profile Locale`	The Verity locale used by the K2 Server to open the Profile Service. This setting overrides the locale specified in the `vdksettingset` command. See "vdksettingset" on page 73. Each Profile Service can use only one locale. The selected locale is relevant to queries, but not documents. The default is `uni`. NOTE: Do not modify the locale once it is set. See the *Verity Locale Configuration Guide* for acceptable values.

Argument	Description
`Enter a Styleset alias for the following Gateways:`	This section defines the styleset used by the Profile Service when a user elects to profile a document with a particular gateway. To view a list of available stylesets, use the `styleget` command. See "styleget" on page 202. When you insert a Profile Service, the specified style files are copied from the directory *`dataDir`*`/stylesets` to the directory *`dataDir`*`/services/`*`serveralias`*`/profstyles/`*`profilealias`*`/`*`gatewayname`* where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX), and *`gatewayname`* is one of `vgwnotes`, `vgwodbc`, `vgwhttp`, `vgwfsys`, `vgwdctm`, or `vgwmsxch`. When you modify a styleset alias for a gateway, you will be asked whether you want to apply the latest version of the styleset. If you enter `y`, the new styleset files will overwrite the existing styleset files.
`File System`	The styleset used by the Profile Service when a user elects to profile a document with the File System gateway.
`HTTP`	The styleset used by the Profile Service when a user elects to profile a document with the HTTP gateway.
`Lotus Notes`	The styleset used by the Profile Service when a user elects to profile a document with the Lotus Notes gateway.
`Documentum`	The styleset used by the Profile Service when a user elects to profile a document with the Documentum gateway.
`ODBC`	The styleset used by the Profile Service when a user elects to profile a document with the ODBC gateway.
`Exchange`	The styleset used by the Profile Service when a user elects to profile a document with the Exchange gateway.

profileget

The `profileget` command enables you to view information about a Profile Service on a K2 Server.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service. You can determine the aliases of Profile Services by using the `hierarchyview` command and choosing to view information for K2 Servers. See "hierarchyview" on page 38.

Output

`rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the Profile Service.
`Threads`	The number of threads used by this Profile Service.
`Locale`	The Verity locale used by the K2 Server to open the Profile Service. Each Profile Service can use only one locale. The selected locale is relevant to queries, but not documents. See the *Verity Locale Configuration Guide* for acceptable values.
`Topic Set Path [Deprecated]`	This prompt was deprecated in K2 V6.0. The topic set is defined by the topic set's alias.
`Topic Set Alias`	The topic set used by this Profile Service.
`Query Parser`	The default query parser used by the Profile Service. This default is used if another query parser is not specified when queries are added.

Argument	Description
`Style (i/n):`	Styleset info information for each gateway through which documents can be evaluated.
`Alias`	The name of the styleset used by this gateway.
`Gateway`	The gateway for the corresponding styleset.
`Profile Net (n)`	The aliases of the Profile Nets available in the Profile Service. Every Profile Service has one or more Profile Nets in it.

profiledel

The `profiledel` command enable you to delete configuration information for a Profile Service. This removes the entries in the `admin`*N*`.xml` file. It does not delete any files.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service.

Configuring a Profile Net

The commands in this section enable you to configure a Profile Net for a K2 Server.

This section describes the following commands:

- profilenetset
- profilenetget
- profilenetdel
- profilenetstateset

profilenetset

The `profilenetset` command enables you to configure Profile Nets for a K2 Server and Profile Service. Before configuring a Profile Net, you must create and configure the Profile Service to which it will be added.

Note If you add or change a Profile Net, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service with which the Profile Net is associated.
`Modify Type`	Indicates whether to modify an existing Profile Net or create a new one. The following options are available: `0` = Update an existing Profile Net. `1` = Insert a new Profile Net.

Argument	Description
`Profile Net:`	
`Alias`	The unique identifier for the Profile Net you want to add or modify.
`Path`	The directory path to the Profile Net. If you do not specify the path, K2 generates it from the `Service Alias`, `Profile Alias` and `Profile Net Alias.`
`Save interval`	At specified intervals, the Profile Net is periodically saved. The default is 300 seconds.
`State`	The state of the Profile Net. The following options are available: `0` = It is offline and is not available. `1` = It is online and is available. `2` = It is hidden. It is not available to users, but can be operated on by administrators. The default is `1`, online.
`Suppress Highlight Info [y/n]?`	Indicates whether terms that match the query are highlighted. If you use encrypted topic sets, set this to `y`. Otherwise, if the query-matching words are highlighted, users can guess the contents of the topic set. The default is `y`, terms are highlighted.

profilenetget

The `profilenetget` command enables you to view information about a Profile Net on a K2 Server.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service with which the Profile Net is associated. You can determine the aliases of Profile Services by using the `hierarchyview` command. See "hierarchyview" on page 38.
`Profile Net Alias`	The unique identifier for the Profile Net. You can determine the aliases of Profile Nets by using either the `hierarchyview` or `profileget` commands. See "hierarchyview" on page 38, and "profileget" on page 138.

Output

For each Profile Net, `rcadmin` shows the following information:

Argument	Description
`Profile Net (`*i*`/`*n*`):`	
`Alias`	The unique identifier for the Profile Net.
`Path`	The directory path to the Profile Net.
`Save interval`	The number of seconds after which K2 automatically saves the Profile Net data. The default is 300 seconds.

Argument	Description
`State`	The state of the Profile Net. The following options are available: `offline` = The Profile Net is offline and is not available. `online` = The Profile Net is online and is available. `hidden` = The Profile Net is hidden. It is not available to users, but can be operated on by administrators.
`Suppress highlight Info`	Indicates whether terms that match the query are highlighted.

profilenetdel

The `profilenetdel` command enables you to delete configuration information for a Profile Net. This removes the entries in the `admin`*N*`.xml` file. It does not delete any files.

Note If you delete a Profile Net, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Note If you delete all the Profile Nets for a Profile Service, the Profile Service is also deleted (since it cannot be used without a Profile Net).

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service with which the Profile Net is associated.
`Profile Net Alias`	The unique identifier for the Profile Net you want to delete.

profilenetstateset

The `profilenetstateset` command enables you to configure the state of an existing Profile Net. You can also set this state when creating a new Profile Net using the `profilenetset` command.

Note If you change the state of a Profile Net, perform a full restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 Server with which the Profile Service is associated.
`Profile Alias`	The unique identifier for the Profile Service with which the Profile Net is associated.
`Profile Net Alias`	The unique identifier for the Profile Net.
`Profile Net State`	The state of the Profile Net. The following options are available: `0` = It is offline and is not available. `1` = It is online and is available. `2` = It is hidden. It is not available to users, but can be operated on by administrators.

5

Managing Logging

This chapter describes the commands that manage status and query logs.

- Introduction
- Managing Status Logs
- Managing Query Logs

Introduction

K2 Administration can generate service logs for a particular service. Service logs are a valuable diagnostic tool, often used by Verity Technical Support to determine how your K2 system can work most efficiently. K2 Administration offers you the following logs for viewing:

- status logs for K2 Brokers, K2 Servers, K2 Ticket Servers, K2 Index Servers, and Administration Servers
- query logs for K2 Brokers and K2 Servers

A *status log* shows diagnostic messages for a particular service.

A *query log* shows user query activity for a particular K2 Broker or K2 Server.

Managing Status Logs

The commands in this section manage K2 status logs. The status log records various diagnostic messages on the status of a K2 service. This is valuable information for determining how your K2 system can work most efficiently. The following types of messages can be reported:

```
Fatal
Error
Warning
Status
Verbose
```

The default logging level for K2 services is `Status`. Log levels are cumulative. For example, if the `Status` log level is set, messages for the first four levels are logged (fatal, error, warning, and status). The status log file is stored in the directory *`dataDir`*`/services/`*`servername`*`/log`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

Note If you change the reporting level, perform a quick restart on the K2 service using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

This section describes the following commands:

- statuslogset
- statuslogget

statuslogset

The `statuslogset` command enables you to configure the type of messages recorded in status logs.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service for which you setting logging levels. `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.
`Nominal size (KB, 0 = default 300K)`	The nominal status log file size, specified in KB. The default is 300. If you enter `0` (zero), K2 uses the default (300 KB). When the log exceeds this size, entries are deleted to maintain this size.
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the Status level, error messages for the first four levels are displayed, and so on.

Example

```
rcadmin> statuslogset
Service Alias:myhost_indexserver1
Service Type For "myhost_indexserver1"
[(a)dmin|(b)roker|(s)erver|(t)icket|(i)ndexServer]:i
Nominal size (KB, 0 = default 300K)(0):
Verboseness (Fatal=0, Error=1, Warning=2, Status=3 Verbose=4)(0):4
Save changes? [y|n]:y
<<Return>> SUCCESS

rcadmin> servicesignal
Service Alias:myhost_indexserver1
Type of signal (QuickRestart=0,FullRestart=1,Shutdown=2,Kill=3):0
Save changes? [y|n]:y
<<Return>> SUCCESS
```

statuslogget

The `statuslogget` command enables you to view information about K2 status logs. These logs record the status of K2 services.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type for` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Nominal Size`	The size (in kilobytes) of the log. If you enter `0` (zero), K2 uses the default (300 KB).

`Verboseness`	The level of detail in the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

Managing Query Logs

The commands in this section enable you to specify which user query information is logged for a K2 Broker or K2 Server. By default, K2 enables all query log settings for centralized reporting. However, for a single service you may want to reduce the number of selected settings. This is useful when you want to narrow the query log data for a particular K2 Broker or K2 Server. For example, you may want the query log to show only the IDs of queried categories for a particular K2 Server.

This section describes the following commands:

- querylogset
- querylogget
- querylogdel
- querylogexcludeset
- querylogexcludedel
- querylogexcludeget

querylogset

The `querylogset` command enables you to configure which user query information is recorded in query logs.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 service for which the query log will be configured.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.
`Enable?`	Indicates whether query logging is enabled for this service.

Argument	Description
`Directory Path`	The directory path to the query log. (For example, on Windows, this might be `C:\querylogs\server1`.) This must be a path to an existing directory; it should not point to a file. If the path includes spaces, enclose it in quotation marks. You must create a different directory for every service.
`Roll Over File Size (KBytes, default=1024 KBytes)`	A value, in kilobytes, that represents the maximum file size for the log. Once the file reaches the specified size, K2 starts to write a new log file. The default value is 1024 kilobytes. The actual file size could be larger than the size specified here because only a *complete* line can be written to the file. In XML format, this is a legal XML block. Therefore, the line is written before the file rolls over.
`Query Log Options:`	The prompts in this section define the type of information reported in the query log. Enter `y` to record the specified information in the log. Enter `n` to exclude the information from the log.
`XML File Format?`	If this is enabled, K2 stores the logs in XML format. Otherwise, it stores them in plain text format.
`CollSearch?`	Logs collection search requests.
`TreeSearch?`	Logs knowledge tree search requests.
`ParaSearch?`	Logs parametric index search requests.
`DocStream?`	Logs document streaming requests.
`DocHighlight?`	Logs document highlighting requests. This information includes the document and the query terms to highlight.
`Time?`	Logs the time stamp that shows when the request was processed.
`HitNum?`	Logs the number of hits (occurrences of search terms) from the search.
`SearchTime?`	Logs the total time, in milliseconds, of a search query. This includes the life span of service threads and any time associated with K2 system processing. This setting is related to the `Service Search Time` argument described below.
`Query?`	Logs the user's query in search requests.
`Field?`	Logs the fields used in the search request.

Argument	Description
`Coll?`	Logs the collection used in the search request.
`Tree?`	Logs the knowledge tree used in the search request.
`Category?`	Logs the category ID in the knowledge tree search request. This is a unique identifier for the category.
`Client?`	Logs the IP address of the search client.
`Profile?`	Logs profiling (document evaluation) requests. If this is disabled, then evaluation results are not logged even if `DocInfo`, `PrfName`, or `Results` are enabled. If this is enabled, `DocInfo`, `PrfName`, and `Results` determine what information is logged.
`DocInfo?`	Logs document information. To include this information in the log, the setting `Profile?` must also be enabled.
`PrfName?`	Logs the Profile Net's alias. To include this information in the log, the setting `Profile?` must also be enabled.
`Results?`	Logs evaluation results. To include this information in the log, the setting `Profile?` must also be enabled.
`ServerAlias`	Logs the alias of the K2 Server with which the Profile Net is associated. To include this information in the log, the setting `Profile?` must also be enabled.
`ServiceAlias`	Logs the alias of the Profile Service with which the Profile Net is associated. To include this information in the log, the setting `Profile?` must also be enabled.

Argument	Description
`ServiceSearchTime?`	Logs the life span, in milliseconds, of all service threads used in a search. This does not record the overall K2 processing time. This setting is related to the `Search Time` argument described above. The values for `Search Time` and `Service Search Time` are usually similar. However, a network delay or system load will increase the `Search Time` value. By comparing the `Search Time` and `Service Search Time` values, you can determine whether your K2 system needs tuning.
`Query Suggestions?`	Logs the suggested rewrite of an original user query that was likely misspelled or that yielded little or no hits.
`Query Suggestion Time?`	Logs the total time it takes for K2 to make a spelling suggestion. This value helps determine how spelling suggestions affect search performance.

querylogget

The `querylogget` command enables you to view information about query logs.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service: `b` = K2 Broker. `s` = K2 Server.

Output

`rcadmin` shows the following information:

Argument	Description
`Enable`	Indicates whether the query log is enabled or disabled.
`Directory Path`	The directory path to the query log.
`Roll Over File Size`	The maximum size for the query log (in bytes). When the log is full, K2 overwrites earlier entries.
`Query Log Options:`	The prompts in this section show the type of information reported in the query log.
`XML File Format?`	Logs in XML format. Otherwise, it stores them in plain text format.
`CollSearch?`	Logs collection search requests.
`TreeSearch?`	Logs knowledge tree search requests.
`ParaSearch?`	Logs parametric index search requests.
`DocStream?`	Logs document streaming requests.
`DocHighlight?`	Logs document highlighting requests. This information includes the document and the query terms to highlight.
`Time?`	Logs the time stamp that shows when the request was processed.

Argument	Description
`HitNum?`	Logs the number of hits (occurrences of search terms).
`SearchTime?`	Logs the total time, in milliseconds, of a search query. This includes the life span of service threads and any time associated with K2 system processing. This setting is related to the `Service Search Time` argument described below.
`Query?`	Logs the user's query in search requests.
`Field?`	Logs the fields used in the search request.
`Coll?`	Logs the collection used in the search request.
`Tree?`	Logs the knowledge tree used in the search request.
`Category?`	Logs the ID of the knowledge tree's category used in the search request.
`Client?`	Logs the IP address of the search client.
`Profile?`	Logs profiling (document evaluation) requests. If this is disabled, then evaluation results are not logged even if `DocInfo`, `PrfName`, or `Results` are enabled. If this is enabled, `DocInfo`, `PrfName`, and `Results` determine what information is logged.
`DocInfo?`	Logs document information. To include this information in the log, the setting `Profile?` must also be enabled.
`PrfName?`	Logs the Profile Net's alias. To include this information in the log, the setting `Profile?` must also be enabled.
`Results?`	Logs evaluation results. To include this information in the log, the setting `Profile?` must also be enabled.
`ServerAlias`	Logs the alias of the K2 Server with which the Profile Net is associated. To include this information in the log, the setting `Profile?` must also be enabled.

Argument	Description
`ServiceAlias`	Logs the alias of the Profile Service with which the Profile Net is associated. To include this information in the log, the setting `Profile?` must also be enabled.
`ServiceSearchTime?`	Logs the life span, in milliseconds, of all service threads used in a search. This does not record the overall K2 processing time. This setting is related to the `Search Time` argument described above. The values for `Search Time` and `Service Search Time` are usually similar. However, a network delay or system load will increase the `Search Time` value. By comparing the `Search Time` and `Service Search Time` values, you can determine whether your K2 system needs tuning.
`Query Suggestions`	Logs the suggested rewrite of an original user query that was likely misspelled or that yielded little or no hits.
`Query Suggestion Time`	Logs the total time it takes for K2 to make a spelling suggestion. This value helps determine how spelling suggestions affect search performance.

querylogdel

The `querylogdel` command enables you to delete the configuration information for a query log of a K2 service. This only removes the entries in the `admin`*N*`.xml` file. It does not delete any files.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 service.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.

querylogexcludeset

The `querylogexcludeset` command enables you to configure which machines to exclude from query logging. This command is useful if you use a development or test machine, but you do not want to log queries from it because they would be too numerous.

This command only works if a query log exists. Query logs can be enabled using the `querylogset` command.

Note Using this command overwrites any previous exclusions, so you cannot append items to the list by running the command multiple times. If you need to add multiple machines, use the `Number of IP Filters to Exclude` argument, and add each one separately.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.
`Number of IP Filters to Exclude`	The number of IP addresses to exclude from query logging.
`IP Filter (`*`n`*`/`*`m`*`):`	For each filter specified in `Number of IP Filters to Exclude`, enter values for the following:
`IP address`	The IP address of a machine to exclude (for example `192.168.90.146`).
`Mask`	The subnet mask for the trusted client. This number is ANDed with the IP address to give a range of addresses. For example, with the address above, if the mask is `255.255.0.0` then any addresses starting with `192.168` are accepted. If the mask is `255.255.255.255` then only `192.68.90.146` is accepted.

querylogexcludedel

The `querylogexcludedel` command enables you to remove information about which machines to exclude from query logging.

This command only works if a query log exists. Query logs can be enabled using the `querylogset` command.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.

querylogexcludeget

The `querylogexcludeget` command enables you to view information about which machines are excluded from query logging.

This command only works if a query log exists. Query logs can be enabled using the `querylogset` command.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for this service.
`Service Type For` *`alias`*	The type of service. The following options are available: `b` = K2 Broker. `s` = K2 Server.

Output

If there are no exclusions, `rcadmin` shows the message "`There are no exclusions.`" If there *are* exclusions, `rcadmin` shows the following information for each excluded machine:

Argument	Description
`IP address`	The IP address of a trusted client (for example `192.168.90.146`).
`Mask`	The subnet mask for the trusted client. This number is ANDed with the IP address to give a range of addresses. For example, with the address above, if the mask is `255.255.0.0` then any addresses starting with `192.168` are accepted. If the mask is `255.255.255.255` then only `192.68.90.146` is accepted.

6

Managing Security

This chapter describes the commands that manage security in the K2 system.

- Managing Trusted Clients
- Managing Login Modules
- Managing Persistent Storage

For information on K2 Ticket Servers, which are an integral part of the K2 security system, see "Managing K2 Ticket Servers" on page 75.

Managing Trusted Clients

The commands in this section manage *trusted clients*. A *trusted client* is a computer (IP address) whose client connection is accepted by a service. If an IP address is not a trusted client, its connection is rejected. By default, K2 services accept communications from any IP address, although, a login may be required to perform some or all operations.

If you use trusted clients, you must specify IP addresses for *all* computers that connect to the service, including K2 clients, Administration Servers, K2 Brokers, K2 Servers, K2 Ticket Servers, and K2 Index Servers. If you do not, the excluded services will not connect to your K2 system, and the excluded K2 clients will be unable to access it.

This section describes the following commands:

- trustedclientset
- trustedclientget
- trustedclientdel

trustedclientset

The `trustedclientset` command enables you to add trusted clients to a K2 component.

Note If you add or change a trusted client, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service to which trusted clients will be added.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.
`Number of IP Filters`	The number of IP filters you wish to add.
`IP Filter (`*`n`*`/`*`m`*`):`	For each filter specified in `Number of IP Filters`, enter values for the following:
`IP address`	The IP address of a trusted client (for example `192.168.90.146`).
`Mask`	The subnet mask for the trusted client. This number is ANDed with the IP address to give a range of addresses. For example, with the address above, if the mask is `255.255.0.0` then any addresses starting with `192.168` are accepted. If the mask is `255.255.255.255` then only `192.68.90.146` is accepted.

trustedclientget

The `trustedclientget` command enables you to view information about the IP filters of trusted clients.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the service.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.

Output

If there are no trusted clients set, nothing is shown, If there *are* trusted clients set, `rcadmin` shows the following information for each IP filter:

Argument	Description
`IP Address`	The IP address of a trusted client (for example `192.168.90.146`).
`Mask`	The subnet mask for the trusted client. This number is ANDed with the IP address to give a range of addresses. For example, with the address above, is if the mask is `255.255.0.0` then any addresses starting with `192.168` are accepted. If the mask is `255.255.255.255` then only `192.68.90.146` is accepted.

trustedclientdel

The `trustedclientdel` command enables you to delete configuration information for a trusted client for a K2 service.

Note If you delete a trusted client, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

If a client is connected to a K2 Server, and you remove the IP address of the client from the trusted client list, and perform a quick restart on the server, the client is removed from the list, but is not disconnected from the K2 Server. The client is only disconnected when you perform a full restart, or the client disconnects and tries to reconnect to the server.

Input

Enter the following information when prompted:

Argument	Description
`Service Alias`	The unique identifier for the K2 service for which trusted client information will be deleted.
`Service Type For` *`alias`*	The type of service. The following options are available: `a` = K2 Administration Server. `b` = K2 Broker. `s` = K2 Server. `t` = K2 Ticket Server. `i` = K2 Index Server.

Managing Login Modules

The commands in this section manage the login module a K2 Ticket Server uses for authentication.

See the *Verity K2 Installation and Setup Guide* and the *Verity Security Programming Guide* for more information on login modules.

This section describes the following commands:

- ticketlmset
- ticketlmget

Configuring the Login module

If you do any of the following, you must reconfigure the login module:

- change the Windows domain name.
- change the LDAP schema so that the users or the groups are no longer valid.
- change the login module being used.

To reconfigure the login module:

1. Use the `servicesignal` command to stop the K2 Ticket Server. See "servicesignal" on page 119.
2. Quit `rcadmin`.
3. Delete the Persistent Store module file or clean up the LDAP Persistent Store. This erases all the stored administrator data and index-level security data.
4. Restart the K2 Ticket Server:
   a. Open a DOS or shell window and type

      `k2ticket -alias` *`ticket_server_alias`* `[-charMap` *`charmap`*`]`

      This launches the K2 Ticket Server in a DOS or shell window. (You cannot use `servicesignal` to restart the K2 Ticket Server because `rcadmin` requires the K2 Ticket Server to authenticate the command. So you must restart it manually through a command prompt.)

b. Restart `rcadmin` and login.

   This creates the first administrator.

c. Use the `servicesignal` command to do a full restart of the K2 Ticket Server.

   This runs the K2 Ticket Server as a service instead of running it in the DOS or shell window.

5. Add all other administrators and assign index-level security again.

   You can assign an administrative user in one of three ways:

   - When you log in to the K2 Ticket Server through the K2 Dashboard for the first time, the administrative user is set. The user name and password must be defined on the machine running the K2 Ticket Server, and not on the machine running the K2 Dashboard.
   - In the K2 Dashboard, you can add an administrative user through the **Manage Administrative Users** action.
   - You can use the command-line tool `rctk` to set the administrative user. The syntax is as follows:

     ```
     rctk -port ticketserverportnumber -server ticketservername
     ```

     You then type in `a`. You are prompted for the administrator user name and password.

ticketlmset

The `ticketlmset` command enables you to configure a K2 Ticket Server's login module.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.
`Type`	The type of login module. The following options are available: `1` = The login module uses Windows authentication. `2` = The login module uses UNIX authentication. `3` = The login module uses LDAP authentication. `4` = The login module is user defined.

1. If you chose Windows authentication, enter the following information when prompted:

Argument	Description
`Domain`	The Windows domain (optional). If the user does not provide a domain at log in, the domain specified here is used. If a domain is not specified here, then the domain of the machine running the K2 Ticket Server is used.
`Local Login (y|n):`	If this is enabled, then local users are allowed to log in. Otherwise, only domain users are allowed.
`Optimize Local Login? (y|n):`	If this is enabled, then the module attempts to log in users as local users first. If that fails, it attempts to log them in as domain users. If this is disabled, domain users are tried before local users. If the domain is included in the login credentials, no attempt is made to authenticate locally.

Argument	Description
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the `Status` level, error messages for the first four levels are displayed, and so on. If you change the reporting level, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Limit Trusted Domain Enumeration:`	This section allows you to limit the number of Windows domains queried by K2 for operations such as authenticating users, or determining user group membership. By default, K2 queries *all* the trusted domains in a Windows system. For K2 6.0 or greater, this is not recommended; leave the Domain Enumeration fields empty.
`Domain Count (override default)(0):`	The number of Windows domains that should be queried.
`Domain (i/n)`	
`Name():`	The name of the Windows domain.
`Controller():`	The NetBIOS name of the server that is designated as the domain controller. This is optional. It is recommended you leave this blank, and allow K2 to automatically detect the domain controller.

Argument	Description
`User Group Enumeration Type:`	The type of user information retrieved from this domain. The options are: 1 = None. K2 does not retrieve the global or local domain groups to which the user belongs from the Windows domain. 2 = All. K2 retrieves the global and local domain groups to which the user belongs from the Windows domain. 3 = Global. K2 retrieves the global groups to which the user belongs from the Windows domain. 4 = Local. K2 retrieves the local domain groups to which the user belongs from the Windows domain. Currently, memberships in nested global groups in an Active Directory native mode environment are not retrieved.
`User Group Authentication Type:`	This determines how the user's group information from the Windows domain is used in K2. The options are: 1 = None. The group information is not used. 2 = All. The group information is used to provide document-level access and index-level access. See options 3 and 4 for more information. 3 = Document. The group information is used to access documents in *pre-authentication* mode. If pre-authentication is enabled, K2 only requires a valid user name and optional domain name. Authentication is not performed, but the user name and domain are verified against the Windows repository. Pre-authentication must be enabled in the K2 Ticket Server. See "ticketset" on page 76. 4 = Index. The group information is used to access a Verity index.
`Enable Built In Groups? (y|n) (No):`	If this is enabled, the Windows *built-in* groups "everyone," "local," and "authenticated users" are also included in a user's group list. This is set to `N` by default. It is recommended you exclude built-in groups from the group list. If you want to customize the list of built-in groups that are included in the user's group list, set this to `N`, and define the groups at the next prompt.
`Custom Built In Group Count (override default)(0):`	This enables you to customize the list of built-in groups that are included in the user's group list. Defining a built-in group(s) at this prompt overrides the built-in groups specified in the previous prompt.
`Group (i/n) Name():`	The name of the built-in group.

2. If you chose UNIX authentication, enter the following information when prompted:

Argument	Description
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the `Status` level, error messages for the first four levels are displayed, and so on. If you change the reporting level, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

3. If you chose LDAP authentication, enter the following information when prompted:

Argument	Description
`Admin ID`	The the LDAP administrator user ID. This field is optional.
`Admin Password`	The the LDAP administrator password
`sslCertPath`	The path to the SSL certificate used for SSL connections to the LDAP server. This field is optional. This path identifies the location of the `cert7.db` file, which contains a list of CAs (Certificate Authorities) that are considered trusted. Any certificate that an LDAP server returns is checked to ensure a trusted CA has signed that certificate. K2 does not support client (two-sided) authentication. It supports SSL using server-side certificates only.
`Hostname`	The name or IP address of the LDAP server.
`Port`	The port used by the LDAP server.
`Number of Connections`	The maximum number of connections used by the LDAP server.
`Charmap`	Deprecated in K2 V5.0. This value is not used.

Argument	Description
`Look In Group? (y|n)`	Indicates where group membership information is stored. If enabled, a user's group membership information contains a list of user members and is determined by looking at the group. Otherwise, it is assumed that user information will contain a list of associated groups.
`Preauth Mapping DN Key:`	An LDAP attribute used to retrieve a DN that replaces the original user DN. This value is used in the Verity `DN_KEY` in the credential structure.
`Group Information:`	This section configures group information fields.
`Expand Attribute`	Set to **`t`** if you want the group to store the full domain name. When set to **`f`**, the group is stored as the group key only.
`Find Attribute`	The attribute for a user entry, if it is stored in the group record.
`Find DN`	The domain name root for group entries.
`Key`	The directory key for group entries.
`Object Class`	The LDAP object class that stores group information. This should be the base class for all groups. The default value is `groupofuniquenames`.
`User Information:`	This section configures user information fields.
`Expand Attribute`	Set to **`t`** to use the full domain name for the group. When set to **`f`**, the group is stored as the group key only.
`Find Attribute`	The attribute for a user entry, if the group is stored in a user record.
`Find DN`	The the domain name root for user entries.
`Key`	The directory key for user entries.
`Object Class`	The LDAP object class that stores user information. This should be the base class for all users. The default value is `person`.

Argument	Description
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the `Status` level, error messages for the first four levels are displayed, and so on. If you change the reporting level, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

4. If you chose user defined authentication, enter the following information when prompted:

Argument	Description
`Module Name`	The name of the dynamic library containing the user-defined login module. For example, `k2lmtg`, for the sample Ticket Granting login module. Libraries are stored in the following directory: *`install`*`/k2/`*`_platform`*`/bin` where *`install`* represents the Verity installation path. You create the library using the Verity `k2seclm.h` header file which is stored in the following directory: *`install`*`/k2/include`
`Parameter Count`	The total number of authentication parameters required by this module.
`Attribute (`*`n`*`/`*`m`*`):`	For each authentication field name, enter a name and value.
`Name`	The name of a authentication field in the login module. For example, *`mm_name`* for mother's maiden name.
`Value`	The value of the authentication field. For example, `smith`.

ticketlmget

The `ticketlmget` command enables you to view information about a K2 Ticket Server's login module.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.

Output

1. If the login module uses Windows authentication, `rcadmin` shows the following information:

Argument	Description
`Login Module Information:`	
`Service`	`k2lmnt` = The login module uses Windows authentication.
`Domain`	The Windows domain
`Local Login`	If this is enabled, then local users are allowed to log in. Otherwise, only domain users are allowed.
`Optimize Local Login`	If this is enabled, then the module attempts to log in users as local users first. If that fails, it attempts to log them in as domain users. If this is disabled, domain users are tried before local users. If the domain is included in the login credentials, no attempt is made to authenticate locally.

`Verbose`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

Argument	Description
`Enumerated Domains`	If trusted domains were enumerated for the login module, this section lists the Windows domains queried by K2 for operations such as authenticating users, or determining user group membership.
`Domain (i/n):`	
`Name():`	The name of the Windows domain.
`Controller():`	The NetBIOS name of the server that is designated as the domain controller.
`Enumeration Type:`	The type of user information retrieved from this domain. The options are: 1 = None. K2 does not retrieve the global or local domain groups to which the user belongs from the Windows domain. 2 = All. K2 retrieves the global and local domain groups to which the user belongs from the Windows domain. 3 = Global. K2 retrieves the global groups to which the user belongs from the Windows domain. 4 = Local. K2 retrieves the local domain groups to which the user belongs from the Windows domain.
`Authentication Type:`	Indicates how the user's group information from the Windows domain is used in K2. The options are: 1 = None. The group information is not used. 2 = All. The group information is used to provide document-level access and index-level access. See options 3 and 4 for more information. 3 = Document. The group information is used to access documents in *pre-authentication* mode. If pre-authentication is enabled, K2 only requires a valid user name and optional domain name. Authentication is not performed, but the user name and domain are verified against the Windows repository. Pre-authentication must be enabled in the K2 Ticket Server. See "ticketset" on page 76. 4 = Index. The group information is used to access a Verity index.
`Enable Built In Groups`	Indicates whether the Windows *built-in* groups "everyone," ""local," and "authenticated users," or a custom list of built-in groups are included in the user's group list.

2. If the login module uses UNIX authentication, `rcadmin` shows the following information:

Argument	Description
`Login Module Information:`	
`Service`	`k2lmunix` = The login module uses UNIX authentication.

`Verbose`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

3. If the login module uses LDAP authentication, `rcadmin` shows the following information:

Argument	Description
`Login Module Information:`	
`Service`	`k2lmldap` = The login module uses LDAP authentication.
`Admin ID`	The LDAP administrator user ID.
`Admin Password`	The LDAP administrator password.
`sslCertPath`	The path to the SSL certificate used for SSL connections to the LDAP server.
`Hostname`	The name or IP address of the LDAP server.
`Port`	The port used by the LDAP server.
`Connections`	The maximum number of connections used by the LDAP server.
`Charmap`	Deprecated in K2 V5.0. This value is not used.

Argument	Description
`Look In Group`	Indicates where group membership information is stored. If enabled, a user's group membership information contains a list of user members and is determined by looking at the group. Otherwise, it is assumed that user information will contain a list of associated groups.
`Preauth DN Key:`	The LDAP attribute used to retrieve a DN that replaces the original user DN. This value is used in the Verity `DN_KEY` in the credential structure.
`Groupinfo:`	
`Expand Attribute`	If set, the group stores the full domain name. When disabled, the group is stored as the group key only.
`Find Attribute`	The attribute for a user entry, if it is stored in the group record.
`FindDN`	The domain name root for group entries.
`Key`	The directory key for group entries.
`Object Class`	The LDAP object class that stores group information. This should be the base class for all groups.
`Userinfo:`	
`Expand Attribute`	If set, the full domain name is used for the group. When disabled, the group is stored as the group key only.
`Find Attribute`	The attribute for a user entry, if the group is stored in a user record.
`FindDN`	The the domain name root for user entries.
`Key`	The directory key for user entries.

Argument	Description
`Objectclass`	The LDAP object class that stores user information. This should be the base class for all users.

`Verbose`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

4. If the login module is user defined, `rcadmin` shows the following information:

Argument	Description
`Login Module Information:`	
`Module Name`	The name of the dynamic library containing the user-defined login module.
`Service`	`k2luserdefined` = The login module is user defined.
`Ticket User (`*`n/m`*`):`	For each user-defined attribute pair, `rcadmin` displays the following:
`Name`	The name for a name-value pair.
`Value`	The corresponding value for that `Name`.

Managing Persistent Storage

The commands in this section manage the K2 persistent store module.

The K2 Ticket Server requires a persistent store module to operate. Two persistent store modules are provided: Default PSM (DPSM) and LDAP-based PSM (LDAP PSM). APIs are provided for custom login and persistent store modules.

The default PSM stores the user's credentials in memory only. The default PSM removes the user's credentials if all instances for that user have timed out or logged off. However, a cache credentials setting stores a user's gateway credentials even after the user has logged out of the system. An encrypted file, with the default name cstore.psm, stores the system's administration users and the groups assigned index- or collection-level security.

The LDAP PSM requires a directory server to store the ticket information. The LDAP PSM stores all the data in the directory server, including users' encrypted gateway credentials, groups assigned index- or collection-level security, and users' credentials used for gateway authentication.

See the *Verity K2 Installation and Setup Guide* and the *Verity Security Programming Guide* for more information on the persistent store module.

See the *Verity K2 Dashboard Administrator Guide* for information on configuring an LDAP server to be used as a persistent store by the K2 Ticket Server.

This section describes the following commands:

- ticketpsmset
- ticketpsmget

ticketpsmset

The `ticketpsmset` command enables you to configure a K2 Ticket Server's persistent store module.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.
`Persist Module Type`	The type of the persistent store module. The following options are available: `1` = It uses the default dynamic memory storage scheme. `2` = It uses LDAP storage. `3` = It uses a user defined storage scheme.

1. If you chose the default dynamic memory storage, enter the following information when prompted:

Argument	Description
`Collection Access Cache?`	This value is not currently used.
`Credential Cache?`	If this is enabled, the module keeps credentials in memory after the user logs out. Otherwise, it does not keep them in memory.
`CStore`	The location of the `.psm` file (optional). If this is not specified, the default location is used: *`dataDir`*`/services/ticket`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

Argument	Description
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the `Status` level, error messages for the first four levels are displayed, and so on. If you change the reporting level, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

2. If you chose LDAP storage, enter the following information when prompted.

Note If you modify the data in the LDAP persistent store module, you must restart the K2 Ticket Server before authenticating any more users.

Argument	Description
`Host`	The name or IP address of the LDAP server.
`Port`	The port used by the LDAP server
`LDAP Connection Pool`	The maximum number of connections used by the LDAP server.
`Admin ID`	The LDAP administrator user ID. This field is optional.
`Admin Password`	The LDAP administrator password.
`sslCertPath`	The path to the SSL certificate used for SSL connections to the LDAP server. This field is optional. This path identifies the location of the `cert7.db` file, which contains a list of CAs (Certificate Authorities) that are considered trusted. Any certificate that an LDAP server returns is checked to ensure a trusted CA has signed that certificate. K2 does not support client (two-sided) authentication. It supports SSL using server-side certificates only.

Argument	Description
`psCollSecBaseDN`	The base domain name for collection security that is in the `LDAPinitialization.ldif` file. See *Verity K2 Dashboard Administrator Guide* for more information on the `LDAPinitialization.ldif` file.
`psRepBaseDN`	The base domain name for repository security that is in the `LDAPinitialization.ldif` file. See *Verity K2 Dashboard Administrator Guide* for more information on the `LDAPinitialization.ldif` file.
`Verboseness`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows: `0` = Fatal `1` = Error `2` = Warning `3` = Status `4` = Verbose The default value is `Status`. Log levels are cumulative. For example, if you select the `Verbose` level, error messages for all five levels are displayed. If you select the `Status` level, error messages for the first four levels are displayed, and so on. If you change the reporting level, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

3. If you chose user defined storage, enter the following information when prompted:

Argument	Description
`Module Name`	The dynamic library containing the user-defined persistent store module.
`Parameter Count`	The number of name/value pairs.
`Attribute (n/m):`	For each user defined attribute pair, enter:
`Name`	The name for a name-value pair.
`Value`	The corresponding value for that `Name`.

ticketpsmget

The `ticketpsmget` command enables you to view information about a K2 Ticket Server's persistent store module.

Input

Enter the following information when prompted:

Argument	Description
`Ticket Alias`	The unique identifier for the K2 Ticket Server.

Output

The `Service` field indicates what type of storage scheme is being used.

1. If the persistent store module uses the default dynamic memory storage, `rcadmin` shows the following information:

Argument	Description
`Service`	`k2psmdm` signifies this uses the default dynamic memory storage.
`Collection Cache`	This value is not currently used.
`Credential Cache`	If this is enabled, the module keeps credentials in memory after the user logs out. Otherwise, it does not keep them in memory.
`Cstore`	The location of the `.psm` file (optional). If this is not specified, the default location is used: *`dataDir`*`/services/ticket`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

`Verbose`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

2. If the persistent store module uses LDAP storage, `rcadmin` shows the following information:

Argument	Description
`Service`	`k2psldap` signifies this uses LDAP storage.
`Host`	The alias of the machine hosting this service.
`Port`	The port used by this service.
`LDAP Connection Pool`	The maximum number of connections used by the login module.
`Admin ID`	The user ID of the LDAP administrator.
`Admin Password`	The password of the LDAP administrator.
`sslcert Path`	The path to the SSL certificate used for SSL connections to the LDAP server. This path identifies the location of the `cert7.db` file, which contains a list of CAs (Certificate Authorities) that are considered trusted. Any certificate that an LDAP server returns is checked to ensure a trusted CA has signed that certificate. K2 does not support client (two-sided) authentication. It supports SSL using server-side certificates only.
`Collection SecbaseDN`	The LDAP context where information about index security is stored.
`RepBaseDN`	The LDAP context where user credentials are stored.

`Verbose`	The level of detail of the reporting. The values for logging levels, in increasing detail, are as follows:
	`Fatal`
	`Error`
	`Warning`
	`Status`
	`Verbose`

3. If the persistent store module uses user defined storage, `rcadmin` shows the following information:

Argument	Description
`Module Name`	The name of the persistent store module.
`Service`	`k2userdefined` signifies this uses a user defined storage scheme.
`Ticket User (`*`n`*`/`*`m`*`):`	For each user defined attribute pair:
`Name`	The name for a name-value pair.
`Value`	The corresponding value for that `Name`.

7

Managing Indexes

The commands in this section enable you to manage Verity indexes—collections, knowledge trees, topic sets, parametric indexes, and recommendation indexes.

- Introduction
- Basic Commands
- Managing Collections
- Managing Knowledge Trees
- Managing Topic Sets
- Managing Parametric Indexes
- Managing Recommendation Indexes

Introduction

When you introduce an index to the K2 system—a collection, knowledge tree, parametric index, or recommendation index—you must configure it in two separate ways:

- Register the index with the Administration Server on a host.
- Attach the index to a K2 Server using the `indexattach` command. See "indexattach" on page 194.

When you register an index, you provide an alias for the index and give its location for the host computer. The path you specify must be correct for the host machine. You register an index only once per host. Before the index can be searched, you must attach it to the K2 Server running on the same host.

Unlike a K2 Server, an index does not have to be attached to a K2 Index Server before it can be used.

Topic sets must be registered, but do not have to be attached to a K2 Server.

Registering Indexes

You can register an index with an Administration Server using the `rcadmin` command-line tool, VAdministration API, or K2 Dashboard. To register an index with an Administration Server using `rcadmin`, use the following commands:

Index Type	rcadmin command
Collections	`collset`
	`collget`
	`indexvdkset`
Knowledge Tree	`treeset`
	`treeget`
Topic Sets	`topicsetset`
	`topicsetget`
Parametric Indexes	`paraset`
	`paraget`
Recommendation Indexes	`riset`
	`riget`

Basic Commands

The commands in this section are basic commands that are used for more than one type of Verity index.

This section describes the following commands:

- indexdel
- indexdetach
- indexattach
- indexserverget
- indexstateset
- indexvdkset
- indexviewset
- styleset
- styleget
- styledel

indexdel

The `indexdel` command enables you to delete registration information for a collection, knowledge tree, topic set, parametric index, or recommendation index from the `admin`*N*`.xml` file. It can also delete the index's directory path and files.

Before using this command, you must first detach the index from all K2 Servers above it in the K2 hierarchy, and remove any jobs associated with the index. To detach an index, see "indexdetach" on page 193. To delete a user defined job, see "udjdel" on page 298. To determine whether an index is associated with a job, see "indexview" on page 39.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the index is registered.
`Index Alias`	The unique identifier for the index you want to delete.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree `p` = parametric index `s` = topic set `r` = recommendation index
`Remove the index directory? [y/n]`	Select `y` to delete the index's directory path and files in addition to the registration information. The index's directory path must be defined in the `verity.cfg` with read/write permission before it can be deleted.

indexdetach

The `indexdetach` command enables you to detach a collection, knowledge tree, parametric index, or recommendation index from a K2 Server. Before using the `indexdel` command, you must first use this command to detach the index from all K2 Servers above it in the K2 hierarchy.

Note After detaching an index, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 Server to which the index is attached.
`Index Alias`	The unique identifier for the index you want to detach.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree `p` = parametric index `r` = recommendation index

indexattach

The `indexattach` command enables you to attach a collection, knowledge tree, parametric index, or recommendation index to a K2 Server.

When attaching a recommendation index of type `doc`, you must attach it to the same K2 Server to which its source collection is attached. If the source collection is attached to more than one K2 Server, the recommendation index can also be attached to more than one K2 Server.

Note After attaching an index, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Index Alias`	The unique identifier for the index you want to attach.
`Index Type`	The type index. The following options are available: `c` = collection `t` = knowledge tree `p` = parametric index `r` = recommendation index
`Server Alias`	The unique identifier for the K2 Server to which you want to attach the index.
`Modify Type`	Indicates whether to modify existing information for an attached index, or attach a new index. The following options are available: `0` = Update information for an index. `1` = Register a new index.
`Index State`	The state of the index. The following options are available: `0` = offline and not available. `1` = hidden. The index is not available to users, but can be operated on by administrators. `2` = online and available.

Argument	Description
`Threads (default=3)`	The number of threads used by the index. The default is 3.

indexserverget

The `indexserverget` command enables you to view information for an index on a K2 Server.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 Server to which the index is attached. To show information about all K2 Servers, press the Enter key.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree `p` = parametric index `r` = recommendation index
`Index Alias`	The unique identifier for the index for which you want to view information.

Output

`rcadmin` shows the following information for each K2 Server to which the collection is attached.

Argument	Description
`Server Alias`	The unique identifier for the K2 Server.
`ServerSpec`	The specification for the machine hosting the K2 Server.
`Index Alias`	The unique identifier for the index.
`Index Path`	The directory path to the index.

Argument	Description
`Index State`	The running state of the index. The following options are available: `offline` = offline and not available. `online` = online and available. `hidden` = hidden. It is not available to users, but can be operated on by administrators. `unknown` = A state of "unknown" is usually returned when a collection is attached to a K2 Server, but has not been restarted.
`Configured State`	The configured state of the index according to the `admin`*N*`.xml` file. The following options are available: `offline` = offline and not available. `online` = online and available. `hidden` = hidden. It is not available to users, but can be operated on by administrators. `unknown` = A state of "unknown" is usually returned when a collection has been attached to a K2 Server, but has not been restarted.
`Threads`	The number of threads used by the index.

indexstateset

The `indexstateset` command enables you to change the configured state of an index.

Input

Enter the following information when prompted:

Argument	Description
`Server Alias`	The unique identifier for the K2 Server to which the index is attached.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree `p` = parametric index `r` = recommendation index
`Index Alias`	The unique identifier for the index.
`Index State`	The state to which you want to set the index. The following options are available: `0` = offline and not available. `1` = hidden. It is not available to users, but can be operated on by administrators. `2` = online and available.

indexvdkset

The `indexvdkset` command enables you to configure VDK settings for a collection or knowledge tree.

Note These settings override the defaults set in the `vdksettingset` command, which configures the default VDK settings for a K2 Server. See "vdksettingset" on page 73.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the index is registered.
`Index Alias`	The unique identifier for the collection or knowledge tree.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree
`VDK Setting:`	The prompts in this section correspond to the VDK configuration settings for the K2 Server.
`Allow Stream File?`	Indicates whether VDK is allowed to stream *any* document from the gateway (not only those in the collection). By default, this is set to `n`. Setting it to `y` has security implications because it enables the user to view more documents than usual.
`Max Doc Block Size`	The maximum size (in bytes) of blocks to be read by K2.
`VDK Field Read:`	
`Max Column Size`	The maximum size, in bytes, to read from each field. Make sure this is as large as the largest field you expect to use.
`gw Cert Timeout`	The length of time to wait (in milliseconds) for a certificate from the gateway to expire.
`Date Input Format`	The default date format for all date inputs to K2. For more information on date formats, the *Verity K2 Dashboard Administrator Guide*.

Argument	Description
`Knowledge Base Path`	The directory path to the default knowledge base.
`Topic Set Path`	The directory path to the default topic set.
`Locale`	The Verity locale used by the K2 Server to open the collection or knowledge tree. If no value is specified, the default locale for the K2 Server is assumed. If the locale is already defined in the collection, do not change it using this command. To also set the language used as the default for queries during a VDK session, enter `uni/`*`lang_id`* in this field, where *`lang_id`* is the language identifier for one of the supported languages, such as `fr` (French), or `en` (English). This can only be specified when the locale is the multilanguage (`uni`) locale. See the *Verity Locale Configuration Guide* for acceptable values.
`Charmap`	The character set used by the K2 Server to display results for a collection or knowledge tree. See the *Verity Locale Configuration Guide* for acceptable values.
`Memory Usage:`	
`Max Memory Size`	The maximum memory (in KB) for one K2 Server or K2 Broker instance.
`Max Files`	The maximum number of file handles to hold open for each VDK session.
`Max Memory Percentage`	The maximum amount of memory, in terms of a percentage, for one K2 Server or K2 Broker instance.
`Number of 1K Pages`	The default number of pages (1024 bytes) used by each thread (VDK session) for disk caching of the collection partitions.

indexviewset

The `indexviewset` command enables you to configure Viewing Service settings for a collection or knowledge tree.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the index is attached.
`Index Alias`	The unique identifier for the index.
`Index Type`	The type of index. The following options are available: `c` = collection `t` = knowledge tree
`Index View Detail:`	
`HTML Cache Time`	The amount of time (in seconds) a converted document is kept in the HTML cache before it is marked as "expired." Expired documents are removed from the HTML cache every 8 hours. The default value is 43,200 seconds (which equals 12 hours).
`Number of Sessions`	The number of VDK sessions for this index. The default is to use the number of threads for this index.
`VXmit Block Size`	The size, in bytes, of blocks returned from the Viewing Service. Typical values are 8,000 to 16,000 bytes. The default is 8,000.
`Doc Object Lifetime`	The amount of time (in minutes) a converted document is kept in the document object cache before it is marked as "expired." Expired documents are removed from the document object cache every minute. The default value is 2 minutes.

styleset

The `styleset` command enables you to register a styleset with the Master Administration Server, or modify an existing registration entry.

All stylesets must be stored on the Master Administration Server in the following directory:

`dataDir``/stylesets/`*`styleset_name`*

where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

The *`styleset_name`* directory must exist and should not contain spaces, special characters or multibyte characters. If your styleset is stored on a local host, you must either manually copy the styleset files from the local host to the Master Administration Server, or import the styleset using the K2 Dashboard. See the *Verity K2 Dashboard Administrator Guide* for more information.

Input

Enter the following information when prompted:

Argument	Description
`Modify Type`	Indicates whether to modify existing information for a styleset, or register a new styleset with the Master Administration Server. The following options are available: `0` = Update information for a styleset. `1` = Register a new styleset.
`Style Information:`	
`Alias`	The case-sensitive unique identifier for the styleset.
`Gateway`	The name of the gateway that uses the styleset. For example, `filesys`, `web`, and `odbc`.
`Charmap`	The character set of the style files. If this is left empty, the style files are assumed to be in the internal character set for the locale of the collection with which the style files are used.
`Description`	A text description of the styleset.

Example

```
rcadmin> styleset
Modify Type (Update=0, Insert=1):1
Style Information:
   Alias:my_styleset
   Gateway[(o)dbc|(n)otes|(e)xchange|(d)ocumentum|(f)ilesys|(w)eb|
   o(t)her]:f
   Charmap:1252
   Description:This is a new File System styleset
Save changes? [y|n]:y
<<Return>> SUCCESS
```

styleget

The `styleget` command enables you to view information on the stylesets registered with the Master Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Style Alias (or press enter to list by state):`	The case-sensitive unique identifier for the styleset. To list all stylesets registered with the Master Administration Server, press the `Enter` key at this prompt and at the next prompt. To list stylesets by state, press the `Enter` key at this prompt, and enter a state at the next prompt.
`Style State (store=1,active=2, or press enter to list all styles):`	The state of the styleset(s) you want to view. You only see this prompt if you pressed the `Enter` key at the previous prompt. The following states are available: 1 = Store. The styleset has been created and is ready to use. 2 = Active. The styleset is currently being created. Once the styleset has been created, the state is automatically changes to stored. Press the `Enter` key to view information for all stylesets.

Output

`rcadmin` shows the following information for each styleset.

Argument	Description
`Alias`	The unique identifier for the styleset.
`State`	The state of the styleset.
`Type`	The type of styleset. The following options are available: `New` = The styleset has not been modified since it was created. `Edit` = The styleset has been modified. `Clone` = The styleset is cloned from another styleset. Cloning retrieves all the settings from an existing styleset and sets them in a new styleset. This is useful when you only need to change a few items for a styleset, such as starting points.
`Clone Alias`	The unique identifier for the styleset from which this styleset was cloned.
`Gateway`	The name of the gateway that uses the styleset. For example, `filesys`, `web`, and `odbc`.
`Charmap`	The character set of the style files.
`Description`	A text description of the styleset.
`Path`	The non-default directory path to the styleset on the Master Administration Server.

Example

```
rcadmin> styleget
Style Alias (or press enter to list by state):
Style State (store=1,active=2, or press enter to list all styles):
Style Information(1/4):
    Alias        : Def_FileSystem_Secure
    State        : Store
    Type         : New
    Service Alias:
    Clone Alias  :
    Gateway      : filesys
    Charmap      :
    Description  : default secure File System style files
    Path         :
```

styledel

The `styledel` command enables you to delete registration information for a styleset from the Master Administration Server. This will not remove the styleset directory or files. You must do this manually if necessary.

Input

Enter the following information when prompted:

Argument	Description
`Alias`	The case-sensitive unique identifier for the styleset.

Example

```
rcadmin> styledel
Style Alias:fsp
Save changes? [y|n]:y
<<Return>> SUCCESS
```

Managing Collections

The commands in this section enable you to create, edit or view registration information for collections.

A Verity collection is a 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. For more information about collections, see the *Verity Collection Reference* and the *Verity K2 Dashboard Administrator Guide*.

This section describes the following commands:

- collset
- collcreate
- collget
- collstatget
- collpurge

Adding a Collection to a K2 System

To add a collection to a K2 system, follow these steps:

1. If you have an existing collection, use the `collset` command to register it with the Administration Server, See "collset" on page 206.
2. If the collection does not exist, use the `collcreate` command to create an empty collection, and register the collection. See "collcreate" on page 209.
3. Use the `indexattach` command to attach the collection to the K2 Server. See "indexattach" on page 194.
4. Use the `indexvdkset` command to set any VDK parameters for the collection if required. See "indexvdkset" on page 198.
5. Perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
6. To index the collection, create an indexing job using the `jobcreate` and `jobaddindex` commands. See "Managing Indexing Jobs" on page 264 for more information.

collset

The `collset` command enables you to register a collection with an Administration Server, or modify an existing registration entry. When the collection is registered, the command verifies the collection exists, retrieves the locale from the collection, and writes the locale to the registration entry.

Note When you register collections using a Collection Map file (`.clm`), the collection path is not validated and the locale is not set.

This command does not create collection files. To create collection files *and* register the collection with an Administration Server, use the `collcreate` command. See "collcreate" on page 209.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which you want to register the collection.
`Collection Alias`	The unique identifier for the collection. You can determine the aliases of collections by using the `indexview` command. See "indexview" on page 39.
`Modify Type`	Indicates whether to modify existing information for a collection, or register a new collection with the Administration Server. The following options are available: `0` = Update information for a collection. `1` = Register a new collection.
`Path`	If the collection is stored in a non-default directory, enter the directory path at this prompt. Do not enclose paths in quotation marks. If the path is not valid, an error is returned, and the registration entry is not created.

Argument	Description
`Gateway`	The gateway used in the collection. The following options are available: `o` = ODBC gateway `n` = Lotus Notes gateway `e` = Exchange gateway `d` = Documentum gateway `f` = File System gateway `w` = HTTP gateway `t` = custom gateway To specify a custom gateway, type `t`. You are then prompted for the name of the custom gateway. The custom gateway must already exist.
`Style Alias`	The case-sensitive unique identifier for the *styleset* used by the gateway. The styleset is the complete set of style files. For example, `Def_HTTP`, or `Def_FileSystem`. A style file is 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. To view a list of available stylesets, use the `styleget` command. See "styleget" on page 202.
`Document Access`	The type of access for the collection. The following options are available: `0` = Public. For public collections. The K2 Server does not check credentials through the gateway. `1` = Secure. For secure access. The K2 Server obtains certificates through the gateway, unless the certificates are cached, to perform search and view requests. `2` = Anonymous. For anonymous access (used with the ODBC Gateway only). The K2 Server uses "anonymous" certificates obtained in the server startup.

Argument	Description
`Query Parser`	The name of the query parser used by this collection. The following options are available: `Simple` = Use the standard Verity syntax. `BoolPlus` = Use the explicit Verity syntax. `FreeText` = Use the Verity freetext query parser, which interprets free text using natural language processing tools. `Old FreeText` = Use the old Verity freetext query parser. This value is included for backward compatibility. `Old Simple` = Use the old Verity simple query parser. This value is included for backward compatibility. `Other` = If you select `other`, you are prompted for the name of a custom query parser. The query parser must already exist. If you change this value, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Description`	A text description of the collection.
`Max. Search Time (msecs)`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.

collcreate

The `collcreate` command enables you to create an empty collection. It creates the collection directories and files, *and* registers the collection with the Administration Server. To register an existing collection, use the `collset` command. See "collset" on page 206.

Once the collection is created, it can be added to an existing indexing job using the `jobaddindex` command. See "jobaddindex" on page 282.

Note When you create a collection you must perform a quick restart on the Administration Server using the `servicesignal` command. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Collection Alias`	The unique identifier for the collection.
`Admin Alias`	The unique identifier for the Administration Server with which you want to register the collection.
`Style Alias`	The case-sensitive alias of the *styleset* used by the gateway for this collection. The styleset is the complete set of style files, for example, `Def_HTTP`, or `Def_FileSystem`. A style file is 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. To view a list of available stylesets, use the `styleget` command. See "styleget" on page 202.

Argument	Description
`Collection Path`	The non-default directory path in which the collection is created. It is recommended you do *not* specify a path at this prompt. If a path is not specified, the collection is created in the default data directory: *`dataDir`*`/colls/`*`collalias`* where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). If you choose to use a non-default directory, enter the full path to the collection. Do not enclose the path in quotation marks. If the path is not valid, an error is returned, and the collection is not created. You must also add the non-default directory path to the `verity.cfg` file. Path accessibility is identified to K2 Administration through the `verity.cfg` file. If the path to the collection is not identified in `verity.cfg`, `rcadmin` does not have permission to write to the index. For more information see the *Verity K2 Dashboard Administrator Guide*.
`Locale`	The Verity locale used by the collection, such as `uni`. See the *Verity Locale Configuration Guide* for acceptable values.

collget

The `collget` command enables you to view registration information for a collection.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the collection is registered.
`Collection Alias`	The unique identifier for the collection. To get information about all collections on the server, do not enter an alias at this prompt. You can determine the aliases of collections by using the `indexview` command. See "indexview" on page 39.

Output

`rcadmin` shows the following information for each collection:

Argument	Description
`Alias`	The unique identifier for the collection.
`Configured Path`	The directory path to the collection.
`Gateway`	The gateway used in this collection, for example, `filesys` (File System gateway), `web` (HTTP gateway), or `notes` (Lotus Notes gateway).
`Style Alias`	The unique identifier for the *styleset* used by the gateway. The styleset is the complete set of style files. A style file is 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.
`Description`	A text description of the collection.

Argument	Description
`Document Access`	The type of access for the collection. The following options are available: `0` = Public. For public collections. The K2 Server does not check credentials through the gateway. `1` = Secure. For secure access. The K2 Server obtains certificates through the gateway, unless the certificates are cached, to perform search and view requests. `2` = Anonymous. For anonymous access (used with the ODBC Gateway only). The K2 Server uses "anonymous" certificates obtained in the server startup.
`Query Parser`	The name of the query parser for this collection.
`Max. Search Time`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.
`VDK settings:`	The fields in this section correspond to the VDK configuration settings for the collection. They are set through the `indexvdkset` command. See "indexvdkset" on page 198.
`VDK Stream File`	Indicates whether VDK is allowed to stream *any* document from the gateway (not only those in the collection). By default, this is set to `n`. Setting it to `y` has security implications because it enables the user to view more documents than usual.
`MaxDocBlockSize`	The maximum size (in bytes) of blocks to be read by K2.
`VDK Field Read:`	
`Max Column Size`	The maximum size, in bytes, to read from each field. Make sure this is as large as the largest field you expect to use.
`gwCertTimeOut`	The length of time to wait (in milliseconds) for certificate from the gateway before the command times out.
`DateInputFormat`	The format for parsing dates. See the *Verity K2 Dashboard Administrator Guide*.
`KnowledgeBase`	The directory path to the knowledge base.
`Topic Set Path`	The directory path to the default topic set.

Argument	Description
`Locale`	The Verity locale used by the K2 Server to open this collection. This is either retrieved from the collection when the collection is registered, or is set by the `indexvdkset` command. See "indexvdkset" on page 198.
`Charmap`	The character set used by the K2 Server to display results for the collection.
`Memory Usage:`	
`Max Memory size`	The global maximum memory (in KB) for all collections, unless a value is specified in the particular collection.
`Max files`	The maximum number of file handles that can be opened by a specific search thread.
`Max Memory Percentage`	The maximum amount of memory, in terms of a percentage, that this server can use.
`Number of 1K Pages`	The global value for the number of pages (1024 bytes) for disk caching of the collection partitions, unless a value is specified in the particular collection.
`Collection View Detail:`	The fields in this section correspond to the Viewing Service settings for a collection. They are set through the `indexviewset` command. See "indexviewset" on page 200.
`Number of Sessions`	The number of VDK sessions for this index. The default is to use the number of threads for this index
`VXmit Block Size`	The size, in bytes, of blocks returned from the Viewing Service. Typical values are 8,000 to 16,000 bytes.
`HTML Cache Time`	How long (in seconds) a converted document is kept in the HTML cache before it is marked as "expired." Expired documents are removed from the HTML cache every 8 hours. The default value is 43,200 seconds (which equals 12 hours).
`Document Object Cache Time`	How long (in minutes) a converted document is kept in the document object cache before it is marked as "expired." Expired documents are removed from the document object cache every minute. The default value is 2 minutes.

collstatget

The `collstatget` command enables you to view status information for a collection that is either registered or not registered with an Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`Collection Alias`	If the collection is registered with the Administration Server, enter the unique identifier for the collection.You can determine the aliases of collections by using the `indexview` command. See "indexview" on page 39. To view information for all collections registered with the Administration Server, do not enter an alias at this prompt, or a collection path at the next prompt.
`Collection Path`	If the collection is not registered with the Administration Server, enter the directory path to the collection. Otherwise, press the `Enter` key.
`Skip DDD and DID Size?: (y|n) (No):`	Indicates whether the size (in kbytes) of the collection's document table files (`.ddd`) and document index files (`.did`) are retrieved. If you exclude the files' sizes from the output, the statistics are returned more quickly. If set to `yes`, the files' sizes are not retrieved. The default is `No`. The DDD file stores the document fields, and the DID file stores the document word list for a partition. These files are stored in a collection's `parts` directory.

Output

`rcadmin` shows the following information for each collection:

Argument	Description
`Collection (`*`i`*`/`*`n`*`):`	
`Alias`	The unique identifier for the collection.
`Path`	The directory path to the collection.

Argument	Description
`Date Created`	The date the collection was created.
`Date Last Modified`	The date (`yyyy-mm-dd hr:min:sec`) the collection was last modified.
`Date Last Squeezed`	The date (`yyyy-mm-dd hr:min:sec`) the collection was last optimized using the `mkvdk -optimize squeeze` command. Squeezing a collection removes deleted documents from the collection, which recovers space and improves search performance.
`Date Last Purged`	The date (`yyyy-mm-dd hr:min:sec`) the collection was last purged using the `mkvdk -purge` command. Purging a collection deletes all documents from a collection, but does not delete the collection itself.
`Locale`	The Verity locale used by the collection.
`Charmap`	The character set used by the collection.
`Number of Partitions`	The number of partitions in the collection.
`Number of Live Documents`	The number of active documents in the collection.
`Number of Deleted Documents`	The number of documents that have been deleted from the collection since it was created.
`Size of DDD:`	The size of the collection's document table files (`.ddd`) in kilobytes. The DDD file stores the document fields for a partition. It is stored in a collection's `parts` directory. The combined size of the DDD and DID provides an estimate of the total size of the collection.
`Size of DID:`	The size of the collection's document index files (`.did`) in kilobytes. The DID file stores the document word list for a partition. It is stored in a collection's `parts` directory. The combined size of the DDD and DID provides an estimate of the total size of the collection.

Example

```
rcadmin> collstatget
Admin Alias:marketing_server
Collection Alias:
Collection Path:C:\Program Files\Verity\Data\colls\marketing_coll
Skip DDD and DID Size?: (y|n) (No):
Collection (1/1):
  Alias              :
  Path         :C:\Program Files\Verity\Data\colls\marketing_coll
  Date Created       : 18-Nov-2003 10:50:16 am
  Date Last Modified : 18-Nov-2003 11:45:12 am
  Date Last Squeezed : 18-Nov-2003 11:41:13 am
  Date Last Purged   :
  Locale             : uni
  Charmap            : 1252
  Number of Partitions        : 1
  Number of Live Documents    : 5659
  Number of Deleted Documents : 463
  Size of DDD                 : 32510 kbytes
  Size of DID                 : 80304 kbytes
```

collpurge

The `collpurge` command enables you to remove 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 must purge the other collections in the job using the `collpurge` command, and purge the job itself using the `jobpurge` command, to properly reset it for indexing. See "jobpurge" on page 280.

Note Before purging a collection, it must be offline, and any jobs associated with the collection must not be running.

Input

Enter the following information when prompted:

Argument	Description
`Collection Alias`	The unique identifier for the collection you want to purge.You can determine the aliases of collections by using the `indexview` command. See "indexview" on page 39.
`Admin Alias`	The unique identifier for the Administration Server with which the collection is associated.

Example

```
rcadmin> collpurge
Collection alias:marketing_coll
Admin Alias:marketing_server
Save changes? [y|n]:y
```

Managing Knowledge Trees

The commands in this section enable you to create, edit, or view registration information for knowledge trees.

A *knowledge tree* is a hierarchical data structure that applies categories to documents indexed in a Verity collection. These categories organize documents into subjects of interest. A knowledge tree consists of a content hierarchy—or taxonomy—that provides categories the user can browse, and databases that store the relationship of documents to these categories.

More than one knowledge tree can be associated with a collection. This allows your users to browse different views of the same document set. For example, one knowledge tree can organize a collection by customer, while another can organize the same collection by product name.

For more information on creating knowledge trees, see the *Verity Intelligent Classification Guide*.

This section describes the following commands:

- treeset
- treeget

Adding a Knowledge Tree to a K2 System

To add a knowledge tree to a K2 system, follow these steps:

1. Use the `treeset` command to register the knowledge tree with the Administration Server. See "treeset" on page 219.
2. Use the `indexattach` command to attach the knowledge tree to a K2 Server. See "indexattach" on page 194.
3. Use the `indexvdkset` command to set the locale and character set. See "indexvdkset" on page 198.
4. Perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

treeset

The `treeset` command enables you to register a knowledge tree with an Administration Server, or modify an existing registration entry.

Note To set the default Verity locale and character set to use to open and view knowledge trees, use the `vdksettingset` command. See "vdksettingset" on page 73.

To set a locale or character set for an individual knowledge tree, use the `indexvdkset` command. See "indexvdkset" on page 198. See the *Verity Locale Configuration Guide* for acceptable values.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which you want to register the knowledge tree.
`Alias`	The unique identifier for the knowledge tree.
`Modify Type`	Indicates whether to modify existing information for a knowledge tree, or register a new knowledge tree with the Administration Server. `0` = Update existing information for a knowledge tree. `1` = Register a new knowledge tree.
`Path`	The non-default directory path to the knowledge tree. If this is not specified, the knowledge tree is assumed to be in the default directory *`dataDir`*`/ktrees/`*`ktreealias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

Argument	Description
`Document Access`	The type of access. The following options are available: `0` = Public. For public collections. The K2 Server does not check credentials through the gateway. `1` = Secure. For secure access. The K2 Server obtains certificates through the gateway, unless the certificates are cached, to perform search and view requests. `2` = Anonymous. For anonymous access (used with the ODBC Gateway only). The K2 Server users "anonymous" certificates obtained in the server startup.
`Query Parser`	The name of the query parser used by this knowledge tree. The following options are available: `Simple` = Use the standard Verity syntax. `BoolPlus` = Use the explicit Verity syntax. `FreeText` = Use the Verity freetext query parser, which interprets free text using natural language processing tools. `Old FreeText` = Use the old Verity freetext query parser. This value is included for backward compatibility. `Old Simple` = Use the old Verity simple query parser. This value is included for backward compatibility. `Other` = If you select `other`, you are prompted for the name of a custom query parser. The query parser must already exist. If you change this value, perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
`Description`	A text description of the knowledge tree.
`Max. Search Time (msecs)`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.

treeget

The `treeget` command enables you to view registration information for a knowledge tree.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which this knowledge tree is registered.
`Tree Alias`	The unique identifier for the knowledge tree. To display information about all knowledge trees on the server, press Enter.

Output

For each knowledge tree, `rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the knowledge tree.
`Path`	The non-default directory path to the knowledge tree. If a path is not listed, the knowledge tree is in the default directory *`dataDir`*`/ktrees/`*`ktreealias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).
`Description`	A text description of the knowledge tree.
`Document Access`	The type of access. The following options are available: `0` = Public. For public knowledge trees. The K2 Server does not check credentials through the gateway. `1` = Secure. For secure access. The K2 Server obtains certificates through the gateway, unless the certificates are cached, to perform search and view requests. `2` = Anonymous. For anonymous access (used with the ODBC Gateway only). The K2 Server uses "anonymous" certificates obtained in the server startup.
`Query Parser`	The name of the query parser used by this knowledge tree.

Argument	Description
`Max. Search Time`	The maximum search time, or time to live (TTL), taken by a K2 search request. If the search takes longer than this time, K2 stops the search. The value must be 1000 milliseconds or higher. If this value is zero, a maximum search time is not set.
`VDK settings:`	The fields in this section correspond to the VDK configuration settings for the knowledge tree. They are set through the `indexvdkset` command. See "indexvdkset" on page 198.
`VDK Stream File`	Indicates whether VDK is allowed to stream *any* document from the gateway (not only those in the collection). By default, this is set to `n`. Setting it to `y` has security implications because it enables the user to view more documents than usual.
`MaxDocBlockSize`	The maximum size (in bytes) of blocks to be read by K2.
`VDK Field Read:`	
`Max Column Size`	The maximum size, in bytes, to read from each field. Make sure this is as large as the largest field you expect to use.
`gwCertTimeOut`	The length of time to wait (in milliseconds) for certificate from the gateway before the command times out.
`DateInputFormat`	The format for parsing dates. See the "Date Formats" appendix in the *Verity K2 Dashboard Administrator Guide*.
`KnowledgeBase`	The directory path to the knowledge base.
`Topic Set Path`	The directory path to the topic set.
`Locale`	The Verity locale used by the K2 Server to open this knowledge tree.
`Charmap`	The character set used by K2 Server to display results for this knowledge tree.
`Memory Usage:`	
`Max Memory Size`	The global maximum memory (in KB) for all collections, unless a value is specified in the particular collection.
`Max Files`	The maximum number of file handles that can be opened by a specific search thread.

Argument	Description
`Max Memory Percentage`	The maximum amount of memory, in terms of a percentage, that this server can use.
`Number of 1K Pages`	The global value for the number of pages (1024 bytes) for disk caching of the collection partitions, unless a value is specified in the particular collection.
`Tree View Detail:`	The fields in this section correspond to the Viewing Service settings for a knowledge tree. They are set through the `indexviewset` command. See "indexviewset" on page 200.
`Number of Sessions`	The number of VDK sessions for this knowledge tree. The default is to use the number of threads for this knowledge tree.
`Vxmit Block Size`	The size, in bytes, of blocks returned from the Viewing Service. Typical values are 8,000 to 16,000 bytes
`HTML Cache Time`	How long (in seconds) a converted document is kept in the HTML cache before it is marked as "expired." Expired documents are removed from the HTML cache every 8 hours. The default value is 43,200 seconds (which equals 12 hours).
`Document Object Cache Time`	How long (in minutes) a converted document is kept in the document object cache before it is marked as "expired." Expired documents are removed from the document object cache every minute. The default value is 2 minutes.

Managing Topic Sets

The commands in this section enable you to create, edit or view registration information for topic sets, and to associate a topic set with a collection.

A *topic set* is a grouping of topics that have been compiled for use by a Verity application. A *topic* is a stored query expression that is written in the Verity Query Language (VQL) and is used for classifying documents in a collection. The primary tool for creating and managing topic sets is the Verity Business Console. This product, separate from the K2 Dashboard, is the recommended tool for creating topic sets.

For more information on topic sets, see the *Verity Intelligent Classification Guide* and *Verity Organization Developer's Kit Programming Guide*.

This section describes the following commands:

- topicsetset
- topicsetget
- topicsetcollattach
- topicsetcolldetach

topicsetset

The `topicsetset` command enables you to register a topic set with an Administration Server, or modify an existing registration entry. In addition, if you are registering a topic set that does not already exist, you can create an empty topic set.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which you want to register the topic set.
`TopicSet Alias`	The unique identifier for the topic set you want to import or create. K2 supports multi-byte alias names for indexes.
`Modify Type`	Indicates whether to modify existing information for a topic set, or register a new topic set with the Administration Server. The following options are available: `0` = Update existing information for a topic set. `1` = Register a new topic set.
`Path`	The non-default directory path in which the topic set is created. It is recommended you do *not* specify a path at this prompt. If a path is not specified, the topic set is created in the default data directory, *`dataDir`*`/topicsets/`*`topisetalias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). If you choose to use a non-default directory, enter the full path to the topic set. Do not enclose the path in quotation marks. If the path is not valid, an error is returned, and the topic set is not created. You must also add the non-default directory path to the `verity.cfg` file. Path accessibility is identified to K2 Administration through the `verity.cfg` file. If the path to the topic set is not identified in `verity.cfg`, `rcadmin` does not have permission to write to it. For more information see the *Verity K2 Dashboard Administrator Guide*.
`Description`	A brief text description of the topic set. This field is optional.
`Create? [y/n]`	If you are creating registration information for a topic set that does not already exist, you see this prompt. Type `y` to create an empty topic set that can be later populated in the Business Console.

topicsetget

The `topicsetget` command enables you to view registration information for a topic set.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the topic set is registered.
`Topicset Alias`	The unique identifier for the topic set.

Output

For each topic set, `rcadmin` shows the following information:

Argument	Description
`Alias`	The unique identifier for the topic set.
`Path`	The non-default directory path to the topic set. If a path is not listed, the topic set is stored in the default directory *`dataDir`*`/topicsets/`*`topisetalias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).
`Description`	A text description of the topic set.
`Collection:`	
`Alias`	The unique identifier for the collection to which the topic set is attached.

topicsetcollattach

The `topicsetcollattach` command enables you to associate a topic set with a collection.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the collection is registered.
`Collection Alias`	The unique identifier for the collection to which you want to attach the topic set.
`Topicset Alias`	The unique identifier for the topic set you want to attach to the collection.

topicsetcolldetach

The `topicsetcolldetach` command enables you to break the relationship between a topic set and a collection.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server on which the collection resides.
`Collection Alias`	The unique identifier for the collection from which you want to detach the topic set.
`Topicset Alias`	The unique identifier for the topic set you want to detach from the collection.

Managing Parametric Indexes

The commands in this section enable you to create, edit or view registration information for parametric indexes.

A parametric index enables retrieval of documents based on the values of parameters. For more information on parametric indexes, see the *Verity K2 Parametric Developer Guide*.

This section describes the following commands:

- paraset
- paraget
- parastatget

Adding a Parametric Index to a K2 System

To add a parametric index to a K2 system, follow these steps:

1. Use the `paraset` command to register the parametric index with the Administration Server. See "paraset" on page 229.
2. Use the `indexattach` command to attach the parametric index to a K2 Server. See "indexattach" on page 194.
3. Perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.

Note Taxonomies and parametric index trees are stored in parametric indexes, and therefore do not need to be registered separately.

paraset

The `paraset` command enables you to register a parametric index with an Administration Server, or modify an existing registration entry. You can also create an empty parametric index when you register an index using this command. The indexes are created and maintained by the command-line tool `mkpi` or the K2 Dashboard.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which you want to register the parametric index.
`Alias`	The unique identifier for the parametric index.
`Modify Type`	Indicates whether to modify existing information for parametric index, or register a new parametric index. `0` = Update existing information for a parametric index. `1` = Register a new parametric index.
`Index Type`	The type of index referenced by the parametric index. `c` = collection `n` = None
`Collection Alias`	The unique identifier for the collection the parametric index references.
`Path`	The non-default directory path in which the parametric index is created. It is recommended you do *not* specify a path at this prompt. If a path is not specified, the index is created in the default data directory, *`dataDir`*`/pis/`*`pi_indexalias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). If you choose to use a non-default directory, enter the full path to the index. Do not enclose the path in quotation marks. If the path is not valid, an error is returned, and the index is not created. You must also add the non-default directory path to the `verity.cfg` file. Path accessibility is identified to K2 Administration through the `verity.cfg` file. If the path to the index is not identified in `verity.cfg`, `rcadmin` does not have permission to write to it. For more information see the *Verity K2 Dashboard Administrator Guide*.

Argument	Description
`Description`	A text description of this parametric index.
`Create? [y/n]`	If you are creating registration information for a parametric index that does not already exist, you see this prompt. Type `y` to create an empty index.

Example

```
rcadmin> paraset
Admin Alias:marketing_server
Parametric Alias:marketing_pi
Modify Type (Update=0, Insert=1):1
Index Type [(c)ollection,(n)one]:c
Collection Alias:marketing_coll
Path:C:\data\pis\marketing_pi
Description:parametric index for marketing docs
Create? [y|n] (No):y
Save changes? [y|n]:y
<<Return>> SUCCESS
```

paraget

The `paraget` command enables you to view registration information for a parametric index.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the parametric index is registered.
`Parametric Alias`	The unique identifier for the parametric index.

Output

`rcadmin` shows the following information for each parametric index:

Argument	Description
`Alias`	The unique identifier for the parametric index.
`Path`	The non-default directory path to the parametric index. If a path is not listed, the parametric index is stored in the default directory *`dataDir`*`/pis/`*`pi_indexalias`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).
`Internal Index`	Indicates whether the index is reserved for internal K2 functionality.
`Description`	A text description of the parametric index.
`Collection Alias`	The unique identifier for the collection the parametric index references.

parastatget

The `parastatget` command enables you to view status information for a parametric index that is either registered or not registered with an Administration Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`Parametric Alias`	If the parametric index is registered with the Administration Server, enter the unique identifier for the parametric index.You can determine the aliases of parametric indexes by using the `indexview` command. See "indexview" on page 39. To view information for all parametric indexes registered with the Administration Server, do not enter an alias at this prompt, or a parametric path at the next prompt.
`Parametric Path`	If you are retrieving status information for a parametric index that is not registered with an Administration Server, enter the directory path to the parametric index. Otherwise, press the `Enter` key.
`Skip Partition Size?`	Indicates whether the size (in bytes) of the parametric index's partition files (`0000000`*`n`*`.pi`) is retrieved. If you exclude the file's size from the output, the statistics are returned more quickly. If set to `yes`, the partition's size is not retrieved. The default is `No`. This file is stored in the parametric index's directory.

Output

`rcadmin` shows the following information for each parametric index:

Argument	Description
`Parametric (`*`i`*`/`*`n`*`):`	
`Alias`	The unique identifier for the parametric index.
`Path`	The directory path to the parametric index.
`Version`	The version of the parametric index's outline file. The outline file is an XML document that specifies the structure of the parametric index. It defines bucket set elements, bucket elements, and key fields. For more information, see *Verity K2 Parametric Developer Guide*.

Argument	Description
`Last Update`	The date and time at which parametric index files were most recently changed.
`Locale`	The Verity locale used by the parametric index.
`Charmap`	The character set used by the parametric index.
`Last Document ID`	The last document ID in the parametric index as of the last time the parametric index was updated.
`Number of Partitions`	The number of partitions in the parametric index.
`Size of All Partitions`	The size (in bytes) of the parametric index's partition files (`0000000`*`n`*`.pi`).
`Number of Taxonomies`	The number of taxonomy definitions defined in the parametric index's control file (`control.pi`). If the taxonomy does not have any bucketsets defined, the number is zero.
`Taxonomy Name (`*`i`*`/`*`n`*`)`	The names of the taxonomy definitions contained in the parametric index as specified in the outline file.

Example

```
rcadmin> parastatget
Admin Alias:marketing_server
Parametric Alias:marketing_pi
Parametric Path:
Skip partition Size?: [y|n] (No):
Parametric (1/1):
    Alias           : marketing_pi
    Path            : C:\Program Files\Verity\Data\pis\marketing_pi
    Version         : PI V6.1
    Last Update     : 2003-11-04 22:49:33
    Locale          : uni
    Charmap         : 1252
    Last Document ID        : 12900
    Number of Partitions    : 1
    Size of All Partitions  : 134290 bytes
    Number of Taxonomies    : 0
```

Managing Recommendation Indexes

The commands in this section enable you to create, edit or view registration information for *recommendation indexes*. A recommendation index, or RI, contains a logical grouping of entities—such as documents, users, or queries—created by the *Recommendation Engine*.

The Recommendation Engine is a component of the Verity infrastructure that computes how these entities correlate and provides *recommendations*. Recommendations can be based on the users' previously accessed documents and searches, on their departments and colleagues, on documents their colleagues have accessed, and so on. The purpose of the Recommendation Engine is to connect users with the most relevant information, based on a dynamic input context such as a user's profile, query, and so on, as well as feedback from the user community.

For example, when you use `rcadmin` to create a recommendation index that contains groupings of documents, the Recommendation Engine processes the index data. When a user submits a search query, or when a document is accessed by a user, the Recommendation Engine can provide recommendations of other documents, based on this and past user behavior.

This section demonstrates a select part of the Recommendation Engine feature, that is, how to create and manage recommendation indexes in `rcadmin`. For more information on the Recommendation Engine, see the *Verity K2 Recommendation Engine Guide*.

Commands for Setting Host-Wide Defaults

These commands enable you to set or edit host-wide defaults used by the Recommendation Engine. These defaults are set in the Administration Server, and are then made available to all K2 Servers.

- reset
- reget

Commands for Configuring Recommendation Indexes and Data Sources

These commands enable you to configure specific recommendation indexes and define source collections used to populate the indexes.

- riset
- riget
- risourceset
- risourceget
- risourcedel

Commands for Creating Recommendation Index Types

These commands create recommendation index types, and manage interactions between entities of different types.

- ritypeset
- ritypeget
- ritypedel
- rityperelset
- rityperelget
- rityperelde l

Adding a Recommendation Index to a K2 System

To add a recommendation index to a K2 system, follow these steps:

1. If necessary, use the `ritypeset` command to create a user-defined index type for the recommendation index. See "ritypeset" on page 250. This is only required if the required type does not already exist. The common types `doc`, `user`, and `query` are pre-defined and do not need to be created.
2. Use the `riset` command to register the index with the Administration Server. See "riset" on page 242.
3. Use the `risourceset` command to add a data source for the recommendation index. A *source* is a collection from which the recommendation index can be populated. See "risourceset" on page 245. A recommendation index can be populated by multiple source collections.
4. Use the `indexattach` command to attach the recommendation index to a K2 Server. See "indexattach" on page 194.
5. Perform a quick restart on the K2 Server using the `servicesignal` command. This applies the change to the system. See "servicesignal" on page 119.
6. In some cases, you must also update any recommendation index(es) affected by changes you make. For example, you must update a recommendation index when you add a source, or change the setting for the "maximum features," "stemming," or "profile encryption" options.

   Maximum features and profile encryption are host-wide parameters. These settings are stored on the Administration Server and propagated to the K2 Servers.

To update recommendation indexes, use one of the following methods:

- Update *individual* indexes using the following command:

  ```
  mkre -update  ritype rialias
  ```

  where *ritype* is the index's recommendation type, and *rialias* is the index's alias.

- Update *all* indexes on a host using the recommendation job installed with K2 (`Refresh_Recommendation_job`), or the following command:

  ```
  mkre -update
  ```

Recommendation Index Types

A recommendation index has a unique alias and a specified type. `rcadmin` supports the following recommendation index types:

- `doc`, which contains the profiles of the documents in a collection.
- `user`, which contains the profiles of a set of users on a host.
- `query`, which contains the profiles of user queries.
- user-defined types

In the Recommendation Engine, the recommendation index types `doc`, `user`, and `query` are pre-defined and are treated in a special way:

- The recommendation alias "users" (of type `user`) is reserved for the system-wide default user index and should not be explicitly created by the administrator.
- The `doc` type refers to recommendation indexes associated with collections that already exist on the system. Only one collection can be associated with an index of type `doc`. A `doc` recommendation index and its associated collection always share the same alias.
- The `query` type refers to the transient profiles created at runtime. This type is reserved and cannot be used.

To create a custom recommendation index type, use the following commands:

- "ritypeset" on page 250
- "ritypeget" on page 252
- "ritypedel" on page 253

To change default parameters for an index type, use the following commands:

- "rityperelset" on page 254
- "rityperelget" on page 256
- "rityperelde l" on page 257

reset

The `reset` command enables you to set host-wide defaults used by the Recommendation Engine. These settings are stored on the Administration Server and propagated to the K2 Servers.

If you change any of these settings, use the `servicesignal` command to perform a quick restart on each K2 Servers in the K2 domain to which recommendation indexes are attached. This applies the change to the system. See "servicesignal" on page 119.

If you change the following parameters, you must also *update* all recommendation indexes on the host:

- `Max. Features`
- `Stemming`
- `Profile Decryption`

To update recommendation indexes, use one of the following methods:

- Update *individual* indexes using the following command:

  ```
  mkre -update  ritype rialias
  ```

  where *`ritype`* is the index's recommendation type, and *`rialias`* is the index's alias.

- Update *all* indexes on a host using the recommendation job installed with K2 (`Refresh_Recommendation_job`), or the following command:

  ```
  mkre -update
  ```

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for which you are setting Recommendation Engine defaults.
`Locale`	The default Verity locale used to parse queries contained in transactions. See the *Verity Locale Configuration Guide* for acceptable values.
`Max. features (50)`	The default maximum number of features in all recommendation indexes. This value corresponds to the terms used in a profile, such as `apple*1`, `orange*1`. Valid values are from 1 to 100. The default value is 50. This means each profile can start with 25 terms (the default number of features in `VdkWeightedFeatures`), but can grow to 50 terms. Once the profile reaches 50 terms, features with low weights are removed to create space for features with higher weights. If this value is increased, the accuracy of the recommendation increases, as does the size of the recommendation index. This value can be overridden for a given recommendation index type by defining `Max. Features` on an attached recommendation type using the `ritypeset` command. See "ritypeset" on page 250.
`Score factor (100)`	The default score factor for all recommendation indexes. Valid values are from 1 to 100. The default is 100. The score factor is used to combine scores from other knowledge sources (such as VDK) with recommendation engine scores to produce adaptively ranked search results. For example, if the score factor value is 85, the Recommendation Engine uses the following formula to compute the final score for an adaptive ranking search: 0.85 * RE_score + (1 - 0.85) * VDK_score This value can be overridden for a recommendation type by defining a score factor on an attached recommendation type using the `ritypeset` command. See "ritypeset" on page 250.

Argument	Description
`Profile Decryption? [y|n] (Yes):`	Indicates whether the raw terms in the entity profile are retrieved. If this is set to `y`, actual strings (such as “finance”, and “tax”) are returned when retrieving the entity profile using the API call `VRecommend.getEntity()`. You would set this to y to expose the terms in a profile to an administrator or end user. This allows you to compare terms between entity profiles. If this is set to `n`, the encoded values (such as “12468293”, or “5678234”) for terms are retrieved. To see the raw terms, this setting must be set to `y` before `mkre` creates the recommendation index.
`Stemming? [y|n] (Yes):`	Indicates whether stemming is enabled in the Recommendation Engine. If stemming is enabled, a document is recommended when it contains all the words that share the same stem as the input query. For example, if the query contained the word “house,” all documents whose document profile contained the terms “houses,” “housing,” and “housed” would be recommended.

reget

The `reget` command enables you to view host-wide defaults used by the Recommendation Engine.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for the Recommendation Engine.

Output

`rcadmin` shows the following information:

Argument	Description
`Locale`	The default Verity locale used to parse queries contained in transactions.
`Max. features`	The default maximum number of features in all recommendation indexes. This value corresponds to the terms used in a profile, such as `apple*1`, `orange*1`. For example, if the value is 50, each profile can start with 25 terms (the default number of features in `VdkWeightedFeatures`), but can grow to 50 terms. Once the profile reaches 50 terms, features with low weights are removed to create space for features with higher weights.
`Score factor`	The default score factor for all recommendation indexes.
`Profile Decryption`	Indicates whether the raw terms in the entity profile are retrieved. If this is set to `y`, actual strings (such as “finance”, and “tax”) are returned when retrieving the entity profile using the API call `VRecommend.getEntity()`. If this is set to `n`, the encoded values (such as “12468293”, or “5678234”) for terms are retrieved.

Argument	Description
`Stemming`	Indicates whether stemming is enabled in the Recommendation Engine. If stemming is enabled, a document is recommended when it contains all the words that share the same stem as the input query. For example, if the query contained the word "house," all documents whose document profile contained the terms "houses," "housing," and "housed" would be recommended.
`Available RE Index Types (`*`n`*`):`	If recommendation index types are defined, you see the prompts below:
`Index Type (`*`i`*`/`*`n`*`)`	For each Recommendation Engine index type, `rcadmin` lists the following:
`Max. features`	The default maximum number of features in this recommendation index type. This value corresponds to the terms used in a profile, such as `apple*1`, `orange*1`. For example, if the value is 50, each profile can start with 25 terms (the default number of features in `VdkWeightedFeatures`), but can grow to 50 terms. Once the profile reaches 50 terms, features with low weights are removed to create space for features with higher weights.
`Score factor`	The default score factor for this recommendation index type. This value applies to `doc` index types only.
`History factor`	The history factor determines what percentage of an existing profile (for an index of this type) is preserved when the index is updated. The higher the history factor, the less impact update transactions have on the index, because more of the history is preserved.
`Relevances (`*`n`*`)`	If relevance information is defined for this type, you see this prompt. The `Action`, `Source` and relevance `Value` for the source is shown. Relevance for an index type is set using the `rityperelset` command. See "rityperelset" on page 254.

riset

The `riset` command enables you to register a recommendation index with an Administration Server, or modify an existing registration entry. The indexes are created and maintained by the Recommendation Engine command-line tool `mkre` or the K2 Dashboard.

Once the recommendation index is registered, use the `risourceset` command to add a data source to the index. A *source* is a collection from which the recommendation index can be populated. See "risourceset" on page 245.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the recommendation index is attached.
`RE Index Alias`	The unique identifier for the recommendation index. If this is an RE index of type "doc," the RE Index alias must be the same as the alias of the source collection.
`Modify Type`	Indicates whether to modify existing information for a recommendation index, or register a new recommendation index with the Administration Server. `0` = Update existing information for a recommendation index. `1` = Register a new recommendation index.
`Recommendation Type`	The type of recommendation index. The available types are `doc` and `user`, as well as any user-defined types (for example, "book," "CD," "magazine," or "movie"). The type must already exist. To define user-defined types, use the `ritypeset` command. See "ritypeset" on page 250. A type *must* be specified for a recommendation index.
`Locale`	The Verity locale used by this recommendation index. For the recommendation index of the type `doc`, you must enter the locale of the source collection. See the *Verity Locale Configuration Guide* for acceptable values.

Argument	Description
`Path`	The non-default directory path to the recommendation index. This is optional. If a path is not specified the default directory is used: *`dataDir`*`/ris/`*`ritype`*, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).
`Description`	A short text description of the recommendation index. This is optional.

Example

```
rcadmin> riset
Admin Alias:marketing_server
RE Index Alias:marketing_ri
Modify Type (Update=0, Insert=1):1
Type:doc
Locale:uni
Path:
Description:recommendation index for marketing docs
Save changes? [y|n]:y
<<Return>> SUCCESS
```

riget

The `riget` command enables you to view registration information for the recommendation indexes on the host, including the source details for each recommendation index.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the recommendation index is attached.
`RE Index Alias`	The unique identifier for the recommendation index. To view all recommendation indexes on the host, press the `Enter` key.

Output

`rcadmin` shows the following information for each recommendation index:

Argument	Description
`Available RE Indexes (`*`n`*`)`	
`Index Alias (`*`i`*`/`*`n`*`)`	The unique identifier for this recommendation index.
`Type`	The type of recommendation index.
`Locale`	The Verity locale used by this recommendation index.
`Path`	The directory path to the recommendation index.
`Description`	A text description of the recommendation index.
`Index Sources(`*`n`*`)`	This section lists the sources associated with the recommendation index.
`Alias`	The unique identifier for the source (collection) associated with the index. This source is used to populate the recommendation index.

Argument	Description
`Type`	The source type. Currently, there is only one source type available: collection.
`ID Field`	The field, stored in the source collection, that is used to generate entity IDs in the recommendation index. The default value is the `VdkVgwKey`, also known as the *gateway key*, which is a unique identifier for any document or record in a collection. It is used to locate and retrieve the contents of a document. You can use another VDK field (such as Author, Creator, or From for an index of type user) to derive entity IDs.
`Parameters`	Specifies whether the VDK field `VdkWeightedFeatures` and/or a weighted list of VDK metafields fields from the source collection is used to populate the recommendation profiles of documents in this collection. See "risourceset" on page 245 for more information.

risourceset

The `risourceset` command enables you to add a data source to a recommendation index, or modify information for an existing data source. A *source* is a collection from which the recommendation index can be populated.

If you add or change a source, follow these steps:

1. Use the `servicesignal` command to perform a quick restart on the K2 Servers to which the recommendation index is attached. This applies the change to the system. See "servicesignal" on page 119.
2. Update the recommendation index(es) with which the source is associated. To update recommendation indexes, use one of the following methods:
   - Update *individual* indexes using the following command:

     ```
     mkre -update  ritype rialias
     ```

     where *`ritype`* is the index's recommendation type and *`rialias`* is the index's alias.
   - Update *all* indexes on a host using the recommendation job installed with K2 (`Refresh_Recommendation_job`), or the following command:

     ```
     mkre -update
     ```

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the recommendation index is attached.
`RE Index Alias`	The unique identifier for the recommendation index to which you want to add a source.
`RE Index Source Alias`	The unique identifier for the source (collection) you want to add or modify. This source is used to populate the recommendation index. The source must reside on the same host as the recommendation index. A "user-defined" recommendation index can reference zero or more sources. A "doc" recommendation index can only reference one source; and the referenced source must be attached to the K2 Server. If you are adding a source to an RE index of type "doc," the source alias must be the same as the RE Index alias.
`Modify Type`	Indicates whether to modify information for an existing recommendation source, or add a new source. `0` = Update information for an existing recommendation index. `1` = Register a new recommendation index source.
`ID Field`	The field, stored in the source collection, that is used to generate entity IDs in the recommendation index. The default value is the `VdkVgwKey`, also known as the *gateway key*, which is a unique identifier for any document or record in a collection. It is used to locate and retrieve the contents of a document. You can use another VDK field (such as Author, Creator, or From for an index of type user) to derive entity IDs.
`Parameters`	Specifies whether the VDK field `VdkWeightedFeatures` and/or a weighted list of VDK metafields from the source collection is used to populate the recommendation profiles of entities in this recommendation index. For more information, see "Defining Source Fields" on page 247.

Defining Source Fields

You can specify whether the VDK field `VdkWeightedFeatures` and/or a weighted list of VDK metafields from the source collection is used to populate the recommendation profiles of entities in this recommendation index.

VDK Weighted Features

The internal field `VdkWeightedFeatures` is stored within the collection and contains the 25 most important terms from each document, with a weight for each term. To specify that the `VdkWeightedFeatures` field only is used, type:

```
VdkWeightedFeatures
```

This is the default.

VDK Metafields

To specify that a list of other VDK fields, each with an assigned weight, are used, type:

```
field1*w1,field2*w2,field3*w3,and so on
```

where *`field1`*, *`field2`*, and *`field3`*, are names of VDK fields, and *`w1`*, *`w2`*, and *`w3`* are weights typically in the range 1 to 10.

A high value indicates the metafield will influence recommendations to a greater degree than a low value. K2 calculates the influence according to the number of metafields you define and their relative weights.

Weights are optional. If you enter metafield names only, the weight is evenly divided between all fields entered. For example, if you type

```
subject,author,date
```

K2 calculates the weight of each field as approximately 30 percent.

VDK Weighted Features and VDK Metafields

To specify that `VdkWeightedFeatures` and other VDK fields are used, type:

```
VdkWeightedFeatures*w0,field1*w1,field2*w1, and so on
```

`VdkWeightedFeatures` must be entered first. The weights are optional, and do not affect `VdkWeightedFeatures`. If you use weights, you must provide weights for *all* fields, otherwise, the fields are weighted equally.

Example

```
rcadmin> risourceset
Admin Alias:marketing_server
RE Index Alias:marketing_ri
RE Index Source Alias:marketing_coll
Modify Type (Update=0, Insert=1):1
ID Field:
Parameters:vdkvgwkey*5,author*2,date*3
Save changes? [y|n]:y
<<Return>> SUCCESS
```

risourceget

The `risourceget` command enables you to view registration information for data sources used by a recommendation index. A *source* is a collection from which the recommendation index can be populated.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this recommendation index.
`RE Index Alias`	The unique identifier for the recommendation index.
`RE Index Source Alias`	The unique identifier for the recommendation index source. To view information about all recommendation index sources, press `Enter`.

Output

`rcadmin` shows the following information for each recommendation index:

Argument	Description
`Alias`	The unique identifier for the source (collection) used by the recommendation Index.
`Type`	The source type. Currently, there is only one source type available: collection.
`ID Field`	The field, stored in the source collection, that is used to generate entity IDs in the recommendation index. The default value is the `VdkVgwKey`, also known as the *gateway key*, which is a unique identifier for any document or record in a collection. It is used to locate and retrieve the contents of a document.
`Parameters`	Specifies whether the VDK field `VdkWeightedFeatures` and/or a weighted list of VDK metafields fields from the source collection is used to populate the recommendation profiles of documents in this collection.

risourcedel

The `risourcedel` command enables you to delete registration information for a recommendation index data source. A *source* is a collection from which the recommendation index can be populated.

Note If you delete a recommendation index source, use the `servicesignal` command to perform a quick restart on the K2 Servers to which the recommendation index is attached. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server to which the recommendation index is attached.
`RE Index Alias`	The unique identifier for the recommendation index.
`RE Index Source Alias`	The unique identifier for the source you want to delete. This is optional. If a source is not specified, all sources are deleted.

ritypeset

The `ritypeset` command enables you to create a recommendation index type, or modify an existing type.

If you change any of these settings, use the `servicesignal` command to perform a quick restart on the K2 Servers to which recommendation indexes of this type are attached. This applies the change to the system. See "servicesignal" on page 119.

If you change the `Max. Features` setting for a type, you must also update any recommendation indexes of that type. To update recommendation indexes, use one of the following methods:

- Update *individual* indexes using the following command:

  ```
  mkre -update  ritype rialias
  ```

  where *`ritype`* is the index's recommendation type and *`rialias`* is the index's alias.

- Update *all* indexes on a host using the recommendation job installed with K2 (`Refresh_Recommendation_job`), or the following command:

  ```
  mkre -update
  ```

Input

Enter the following information when prompted:

Argument	Description
`RE Index type name`	The name of the recommendation index type.
`Modify Type`	Indicates whether to modify an existing recommendation index type, or create a new one. `0` = Update an existing recommendation index type. `1` = Create a new recommendation index type.
`Description`	A short text description of the type. This is optional.
`Max. features (default=50):`	The maximum number of features for all recommendation indexes of this type. This value corresponds to the terms used in a profile, such as `apple*1`, `orange*1`. Valid values are from 1 to 100. The default value is 50. This means each profile can start with 25 terms (the default number of features in `VdkWeightedFeatures`), but can grow to 50 terms. Once the profile reaches 50 terms, features with low weights are removed to create space for features with higher weights. If this value is increased, the accuracy of the recommendation increases, as does the size of the recommendation index.
`Score factor (default=100):`	The score factor for this recommendation index type. Valid values are from 1 to 100. The default is 100. This value applies to `doc` index types only. The score factor is used to combine scores from other knowledge sources (such as VDK) with recommendation scores to produce adaptively ranked search results. For example, if the score factor value is 85, the Recommendation Engine uses the following formula to compute the final score for an adaptive ranking search: 0.85 * RE_score + (1 - 0.85) * VDK_score
`History factor (default=100):`	The history factor determines what percentage of an existing profile (for an index of this type) is preserved when the index is updated. The higher the history factor, the less impact update transactions have on the index, because more of the history is preserved. Valid values are from 1 to 100. The default is 100.

Example

```
rcadmin> ritypeset
RE Index type name:archive
Modify Type (Update=0, Insert=1):1
Description:marketing archive
Max. features (default=50):65
Score factor (default=20):85
History factor (default=100):50
Save changes? [y|n]:y
<<Return>> SUCCESS
```

ritypeget

The `ritypeget` command enables you to view registration information for a recommendation index type.

Input

Enter the following information when prompted:

Argument	Description
`RE Index type name`	The name of the recommendation index type. To view information for all recommendation index types, press `Enter`.

Output

`rcadmin` shows the following information for each recommendation index type:

Argument	Description
`RE Index Type (`*i*`/`*n*`)`	The name of the recommendation index type.
`Max. features`	The maximum number of features in this recommendation index type. This value corresponds to the terms used in a profile, such as `apple*1`, `orange*1`. For example, if the value is 50, each profile can start with 25 terms (the default number of features in `VdkWeightedFeatures`), but can grow to 50 terms. Once the profile reaches 50 terms, features with low weights are removed to create space for features with higher weights.

Argument	Description
`Score factor`	The score factor for this recommendation index type. This value applies to `doc` index types only.
`History factor:`	The history factor determines what percentage of an existing profile (for an index of this type) is preserved when the index is updated. The higher the history factor, the less impact update transactions have on the index, because more of the history is preserved.
`Relevances (n)`	If relevance information is defined for this type, you see this prompt. The `Action`, `Source` and relevance `Value` for the source is shown. Relevance for an index type is set using the `rityperelset` command. See "rityperelset" on page 254.

ritypedel

The `ritypedel` command enables you to delete registration information for a custom recommendation index type. You cannot delete the types `doc`, `user`, and `query`, which are automatically configured when you install K2.

When a customized type is currently referenced by a recommendation index on any host in your K2 domain, you cannot delete the type. To delete it, you must first detach and remove the index. See "indexdetach" on page 193 and "indexdel" on page 192.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this recommendation index type.
`RE Index type name`	The name of the recommendation index type.

rityperelset

The `rityperelset` command enables you to configure relevance values for recommendation types. These values are used to determine the default relationships between entities of different types. They define how entities are combined during the recommendation or transaction update process. Specifically, they determine the proportion of input entities used in recommending entities of other types (action=recommendation), and the proportion of input entities used during transaction updates (action=update).

Relevance values for transaction updates can be overridden by setting values for individual transaction calls. However, relevance values for the recommendation process can only be set using this command, and cannot be set in the API. If a relevance value is not defined for a recommendation or transaction update process, then all inputs have equal weight.

The relevance values are defined for each combination of target type and source type.

Note If you change any of these settings, use the `servicesignal` command to perform a quick restart on the K2 Servers to which recommendation indexes of this type are attached. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this recommendation index type.
`RE Index Type`	The type of the recommendation index. This is the target entity of a recommendation or transaction request.
`Source Type`	The type of entity that is used as an input (source) for recommendation and transaction requests.

Argument	Description
`Action [(u)pdate|(r)ecommendation]`	Defines whether the relevance value applies during recommendation calls or during transaction updates. If you are recommending entities of type A (`RE Index Type`) using an entity of type B (`Source Type`) as one of the inputs, then the `Source Value` defines the proportion of all input entities of type B used to create a composite input vector. If you are updating a target entity of type A (`RE Index Type`) with source entities of type B (`Source Type`), then the `Source Value` defines the proportion of all entities of type B used in the modification of the vector for the target entity.
`Modify Type`	Indicates whether to modify existing relevance information for a recommendation index type, or create new relevance information for a recommendation index type. `0` = Update existing information for a recommendation index type. `1` = Create new information for a recommendation index type.
`Source Value`	The proportion applied to the input for the recommendation or update action. The values for a type apply to all recommendation indexes of that type. This value is a percentage in the range 1 to 100. The default for a recommendation action is 50. The default for an update action is 10.

rityperelget

The `rityperelget` command enables you to view relevance information for a recommendation index type. The relevance information determines how entities of the source recommendation type are combined with entities of the target recommendation type to which it is attached during either Update or Recommendation operations.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this recommendation index type.
`RE Index type name`	The name of the target recommendation index type for which relevance information is defined.
`Source type`	The name of the source recommendation type for which the relevance value applies.
`Action [(u)pdate|(r)ecommendation]`	Indicates whether the relevance value is applied during Update or Recommendation actions.

Output

`rcadmin` shows the following information for each index type:

Argument	Description
`Value`	The weighted relevance value assigned to the specified source recommendation type.

rityperelde l

The `rityperelde l` command enables you to delete relevance information for a recommendation index type.

Note If you delete relevance information for a recommendation index type, use the `servicesignal` command to perform a quick restart on the K2 Servers to which recommendation indexes of this type are attached. This applies the change to the system. See "servicesignal" on page 119.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server for this recommendation index type.
`RE Index type name`	The name of the target recommendation type for which relevance information is defined.
`Source type`	The name of the source recommendation type for which the relevance value applies.
`Action [(u)pdate|(r)ecommendation]`	Indicates whether the relevance value is applied during Update or Recommendation operations.

8

Managing Indexing Jobs

This chapter describes the commands that manage K2 Spider indexing jobs.

- Managing K2 Spider
- Managing Indexing Jobs

Managing K2 Spider

The commands in this section manage a K2 Spider Server. K2 Spider gathers documents (crawls), and indexes the documents into a collection.

This section describes the following commands:

- spiderset
- spiderget
- spiderdel

See the *Verity Command-Line Indexing Reference* and the *Verity K2 Dashboard Administrator Guide* for more information on K2 Spider.

Adding a K2 Spider to a Host

To add a K2 Spider, follow these steps:

1. Use the `spiderset` command to register the K2 Spider Server with the Administration Server.

   Note Only one K2 Spider Server can be attached to an Administration Server.

2. Use the `servicesignal` command to start the K2 Spider Server, and add it to the watched service list. An Administration Server monitors watched services, and starts and stops them as necessary. When prompted to enter the service alias of the K2 Spider, enter `@Controller`. See "servicesignal" on page 119.

spiderset

The `spiderset` command enables you to register a K2 Spider with the Administration Server. If you did not install a K2 Spider when you initially installed K2, you can use this command to add a K2 Spider to the host. You can also use this command to change the port number for an Indexer or Controller.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the K2 Spider will be registered.
`Indexer:`	The entries in this section configure the Indexer associated with the K2 Spider. An *Indexer* is a K2 Spider Server process that inserts document data into a collection.
`Alias`	The unique identifier for the Indexer. The default alias that is used when you install a K2 Spider is "indexers."
`Bind`	If an IP address is specified here, then the Indexer is bound to that IP address. Specifying an IP address here is useful if you have more than one network card and wish to only accept requests on one of them.
`Port Start`	The start number of the range of ports used by the Indexer. The default is 9801.
`Port End`	The end number of the range of ports used by the Indexer. The default is 9899.
`Controller:`	The entries in this section configure the Controller associated with the K2 Spider. A Controller manages Crawlers and Indexers, and the gathering and distribution of job information.
`Alias`	The unique identifier for the Controller. The default alias that is used when you install a K2 Spider is "@Controller."
`Bind`	If an IP address is specified here, then the Controller is bound to that IP address. Specifying an IP address here is useful if you have more than one network card and wish to only accept requests on one of them.
`Port`	The port used by the Controller. The default is 9800.

spiderget

The `spiderget` command enables you to view information about an existing K2 Spider Server.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server with which the K2 Spider Server is registered.

Output

`rcadmin` shows the following information:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`Indexer:`	The entries in this section show information about the Indexer associated with the K2 Spider server. An *Indexer* is a K2 Spider Server process that inserts document data into a collection.
`Alias`	The unique identifier for the Indexer.
`Bind`	If an IP address is specified here, then the Indexer is bound to that IP address. Specifying an IP address here is useful if you have more than one network card and wish to only accept requests on one of them.
`Port Start`	The start number of the range of ports used by the Indexer.
`Port End`	The end number of the range of ports used by the Indexer.

Argument	Description
`Controller:`	The entries in this section show information about the Controller associated with the K2 Spider. A *Controller* manages Crawlers and Indexers, and the gathering and distribution of job information.
`Alias`	The unique identifier for the Controller.
`Bind`	If an IP address is specified here, then the Controller is bound to that IP address. Specifying an IP address here is useful if you have more than one network card and wish to only accept requests on one of them.
`Port`	The port used by the Controller.

spiderdel

The `spiderdel` command enables you to delete registration information for a K2 Spider Server from the `admin`*`N`*`.xml` file. It does not delete any files.

Input

Enter the following information when prompted:

Argument	Description
`Admin Alias`	The unique identifier for the Administration Server.
`Indexer Alias`	The unique identifier for the Indexer.
`Controller Alias`	The unique identifier for the Controller. The alias must be `@Controller`.

Managing Indexing Jobs

The commands in this section manage K2 Spider indexing jobs. The indexing process 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. The action of creating (and subsequently running) a job is independent from creating a collection.

To index a collection with `rcadmin`, follow these steps:

1. If the collection does not already exist, use the `collcreate` command to create it. See "collcreate" on page 209. The job will put documents into this collection.
2. Use the `jobcreate` command to create the job. See "jobcreate" on page 267. This involves specifying the indexing starting points and gateway.

   This command also enables you to schedule the job to run either once or at regular intervals in the future, and to create a job chain. See "Creating a Job Chain" on page 265 for more information on job chains.
3. Use the `jobaddindex` command to specify which collection(s) will be populated with documents when you run the job. See "jobaddindex" on page 282.

   Note You can add more than one collection to a job, and more than one job can reference a collection.

4. If you did not set up a schedule to run the job when you created the job, use the `jobstart` command to run the job manually. See "jobstart" on page 278.

This configures a job for K2 Administration. When the job runs, K2 Administration interfaces with K2 Spider.

You can use `rcadmin` to create and run jobs for all Verity gateways. A gateway is a set of access methods, optimized for use by the Verity search engine, used to retrieve a document from a repository for both indexing and viewing. For example, customers use the Exchange gateway to index Microsoft Exchange documents. Gateways exist for the local file system, the web, Documentum, ODBC (Oracle, and so on.), NNTP (NetNews), RTI (Newswire), MAPI (MS Exchange), and Notes. New gateways can be created with the Verity Gateway Developer's Kit (GDK).

Note To configure advanced job options (such as log levels, routing, and crawling parameters) for the HTTP and File System gateways, use K2 Dashboard.

See the *Verity Command-Line Indexing Reference* and the *Verity K2 Dashboard Administrator Guide* for more information on K2 Spider.

This section describes the following commands:

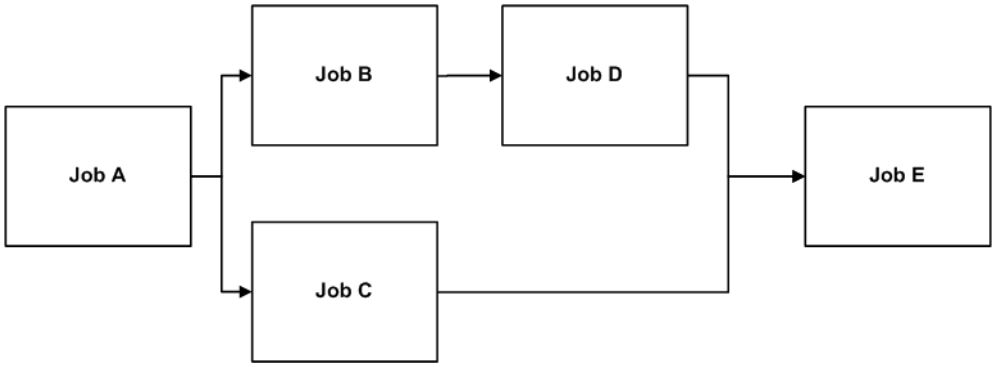
- jobcreate
- jobmodify
- jobgetdetail
- jobgetstats
- jobstart
- jobstop
- jobpause
- jobresume
- jobpurge
- jobdel
- jobaddindex
- jobremoveindex

Creating a Job Chain

You can create a *chain* of jobs using the `jobcreate` or `jobmodify` commands. A chain reaction occurs when you link one job—either a a K2 Spider indexing job or a user defined job—to another. See Chapter 9 for more information on user defined jobs. Each chained job can have one or more prior jobs. When any prior job runs to completion, the job is started. This allows you to set up sequences of jobs. You are not limited in the number of jobs you chain together.

Figure 8-1 shows an example of jobs and predecessors for each.

Figure 8-1 Multiple Chained Predecessors



In this example, the following sequence occurs:

1. Job A runs and finishes.
2. Jobs B and C are triggered and start.
3. When Job B finishes, Job D is triggered and starts.
4. When Job C or D finishes, Job E is triggered and starts.

Notice in step 4 that only one of Job C or D must finish before Job E is triggered to start. In this case, when one of the prior jobs completes, it triggers Job E to start. If the other prior job finishes while Job E is still in a `Running` state, its trigger is queued, as Job E is already underway. When Job E completes, the queued trigger runs Job E a second time. Likewise, if Job E had finished before its second triggering, it would be started again immediately.

jobcreate

The `jobcreate` command enables you to create and schedule K2 Spider indexing jobs. You can also create a job chain with this command. After creating a job, you must associate a collection with the job using the `jobaddindex` command. See "jobaddindex" on page 282.

Note This command enables you to define basic job options. To configure advanced job options (such as log levels, routing, and crawling parameters) for the HTTP and File System gateways, use K2 Dashboard. See the *Verity K2 Dashboard Administrator Guide*.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the job. The job alias must be unique to the host. When you create a job, a number is assigned to it for reference purposes. The first job added is job #1, the second job added is job #2, and so on.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Description`	A brief description of the job you are creating. This is optional.
`Gateway`	The gateway used in the collection. The following options are available: `o` = ODBC gateway `n` = Lotus Notes gateway `e` = Exchange gateway `d` = Documentum gateway `f` = File System gateway `w` = HTTP gateway `t` = custom gateway To specify a custom gateway, type `t`. You are then prompted for the name of the custom gateway. The custom gateway must already exist.
`Number of starting points to index (5 max)`	The total number of starting points for the documents you want to index. The maximum number of starting points is 5.

Argument	Description
`#n:`	The first starting point. For example, enter a file system path or URL (`c:\my_documents`, `\\server\share`, or `http://verity.com`). All starting points for a job must be the same type, that is, file paths or URLs. When you enter a file path, K2 Spider will index all files and subdirectories of that path into the collection. When you enter a URL, K2 Spider initiates a Crawler, which works with an Indexer to gather documents for the job.
`Create job chain? [y|n]`	If you want the completion of another job to trigger the start of this job, type `y`. To view the relationship a job has to other jobs in the chain, use the `jobgetdetail` command. See "jobgetdetail" on page 273. Each chained job must already exist; and all jobs in a chain must be on the same host.
`Number of chained jobs (5 max)`	The number of jobs that can trigger the start of this job. The maximum number of chained jobs is 5.
`Schedule this job?`	Type `y` to create a schedule for the job you are creating. You can create a schedule for a job that is already part of a job chain. If the job is already running because it was started by a predecessor job, then the scheduled start is ignored.
`Frequency`	How frequently the job runs. The available options are: `o` = Runs the job a single time according to the values set in the `Year/Month/Day/Hour/Minute` prompts. `h` = Runs the job hourly according to the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `d` = Runs the job every day at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `w` = Runs the job weekly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `k`= Runs the job on weekdays only, at the time indicated in the `Year/Month/Day/Hour/Minute` prompts `m` = Runs the job monthly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. If you select this option, and the `Day` is `31`, indicating "Monthly on the 31st," the job still runs during months with less than 31 days. In these cases, the job runs on the last day of the month, such as February 28th. `e` = Runs the job monthly, on a particular weekday, such as the second Monday of each month. `i` = Runs the job periodically at specified intervals (minutes).

Argument	Description
`Repeat interval (mins)`	If `Frequency` is `i`, you see this prompt. Enter the interval in minutes at which the job runs. For example, if you set the repeat interval to 300, the job runs every 5 hours. The maximum interval is 32 days (32*24*60-1 = 46079 minutes).
`Start the job now?`	To start the job immediately, type `y`. To start the job at a specified date and time, type `n`. If you type `n`, you see the following prompts.
`Year`	The year that the job starts to run (`yyyy`).
`Month (1-12)`	The month that the job starts to run (1-12).
`Day (1-31)`	If `Frequency` is anything other than `e`, you see this prompt. Enter the day that the job starts to run (1-31).
`Weekday [(M)onday|(T)uesday|(W)ednesday|T(h)ursday|(F)riday|(S)aturday|S(u)nday]`	If `Frequency` is `e`, you see this prompt. Choose the day of the week that the job runs.
`Week of Month [(1)st|(2)nd|(3)rd|(4)th|(l)ast]`	If `Frequency` is `e`, you see this prompt. Choose the week in the month that the job runs.
`Hour (0-23)`	The hour the job starts to run. (0-23)
`Minute (0-59)`	The minute the job starts to run. (0-59)

Example

The example below creates a job named `job1` that indexes documents in the network directories `e:\projects` and `p:\setup`. This job is triggered to start when the job named `chain1` has finished. The job is also scheduled to run the first Monday of every month at 9:30 am, starting in February, 2004.

```
rcadmin> jobcreate
Job Name:job1
Admin Alias:marketing_server
Description:A file system indexing job
Gateway[(o)dbc|(n)otes|(e)xchange|(d)ocumentum|(f)ilesys|(w)eb|
o(t)her]:f
Number of starting points to index (5 max):2
 #1:e:\projects
 #2:p:\setup
Create job chain ? [y|n]:y
Number of chained jobs(5 max):1
Chained job #:
  Job Name:chain1
Schedule this job ? [y|n]:y
Frequency [run(o)nce|(h)ourly|(d)aily|(w)eekly|wee(k)day|
day(m)onthly|w(e)ekdaymonthly|m(i)nute]:e
Start the job now ? [y|n]:n
Year:2004
Month(1-12):2
Weekday [(M)onday|(T)uesday|(W)ednesday|T(h)ursday|(F)riday|
(S)aturday|S(u)nday]:m
Week of Month [(1)st|(2)nd|(3)rd|(4)th|(l)ast]:1
Hour(0-23):9
Minute (0-59):30
Save changes? [y|n]:y
<<Return>> SUCCESS
```

jobmodify

The `jobmodify` command enables you to modify the options for an existing indexing job.

Note This command enables you to modify basic job options. To modify advanced job options (such as log levels, routing, and crawling parameters) for the HTTP and File System gateways, use K2 Dashboard. See the *Verity K2 Dashboard Administrator Guide*.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The case-sensitive name of the job.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Description`	A text description of this job.
`Start Chained Job only on Success? [y|n]`	If you want this job to run only when a prior chained job succeeds, set this to `y`. If you want this job to run when a prior job completes, even if that job fails or aborts, set this to `n`. If you set up multiple prior jobs, and then enable this setting, at least one of the prior jobs must succeed before this job can run.
`Gateway`	The gateway used in the collection. The following options are available: `o` = ODBC gateway `n` = Lotus Notes gateway `e` = Exchange gateway `d` = Documentum gateway `f` = File System gateway `w` = HTTP gateway `t` = custom gateway To specify a custom gateway, type `t`. You are then prompted for the name of the custom gateway. The custom gateway must already exist.

Argument	Description
`(r)emove, (c)hange starting points, or (n)o action?`	To remove the starting points for this job, type `r`. To change the starting points for this job, type `c`. To exit this prompt with no changes, type `n`. If you change the starting points, you are prompted for the total number of starting points, and the new starting points.
`Create job chain? [y|n]:`	If a job chain is not defined for this job, you see this prompt. Type `y` to define a job chain. You are prompted for the number of jobs that trigger this job, and the name of each job. See "jobcreate" on page 267 for information on creating a job chain.
`Number of chained jobs:`*`n`*	If a job chain is already defined for this job, the number of jobs that triggers the start of this job is shown. (Read only.)
`Chained Job #`*`n`*`:` *`JobName, Adminserver`*	This section identifies each job that triggers the start of this job. The prompt shows the job number, job name and the alias of the Administration Server for the job. (Read only.)
`(r)emove, (c)hange job chain, or (n)o Action?`	If a job chain is already defined for this job, you see this prompt. To remove the job chain, type `r`. To change the job chain, type `c`. To exit this prompt with no changes, type `n`. If you are changing the job chain, you are then prompted for the number of chained jobs and the name of each chained job.
`Schedule this Job? [y|n]`	If this job is not currently scheduled, you see this prompt. Type `y` to create a schedule for this job. See "jobcreate" on page 267 for more information on creating a schedule.
`Scheduling information:`	If a schedule is already defined for this job, the mode and frequency of the schedule are shown.
`Mode`	Indicates whether the schedule is enabled or disabled. (Read only.)
`Frequency`	The scheduled job execution frequency. (Read only.)
`(d)isable, (c)hange job schedule, or (n)o Action taken?`	To disable the schedule, type `d`. To change the schedule, type `c`. To exit the prompt with no changes, type `n`.

jobgetdetail

The `jobgetdetail` command enables you to view configuration information for existing indexing jobs on a host.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the job. To view information for all jobs on this host, press the `Enter` key.
`Admin Alias`	The unique identifier for the Administration Server for the job(s).

Output

`rcadmin` shows the following information for each job:

Argument	Description
`Index Job #`*`n`*	The job's number. When you create a job, a number is assigned to it for reference purposes. The first job added is job #1, the second job added is job #2, and so on.
`Job Name`	The job's name.
`Host Alias`	The unique identifier for the host where the job resides.
`Description`	A text description of this job.
`Gateway`	The gateway specified in the job.
`Scheduling information`	
`Mode`	Indicates whether the schedule is enabled or disabled.
`Frequency`	The frequency the job is set to run.
`Start Chained Job only on Success`	Indicates whether this job runs only on the success of a prior job, or if it runs even if prior jobs fail.

Argument	Description
`Chained Jobs that run before this job:`	The number of jobs that will trigger this job when they complete.
`Job #`*`n`*`:` *`jobname, Adminserver`	The chained job(s) that will trigger this job when it completes. The prompt shows the job number, job name and the alias of the Administration Server for the job.
`Chained Jobs that run after this job`	The number of jobs that are started after the completion of this job.
`Job #`*`n`*`:` *`jobname, Adminserver`	The chained job that this job will trigger on its completion. The prompt shows the job number, job name and the alias of the Administration Server for the job.
`Collections:`	This section lists the collections associated with this job.
`Alias:` *`collname, Adminserver`*	The alias of the collection, and the Administration Server with which the collection is registered.
`Sites to index:` *`startingpoints`*	The starting points for this job. For example, a file system path or URL.

jobgetstats

The `jobgetstats` command enables you to view statistics on an indexing job.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name. To view status information for all jobs on this host, press the `Enter` key.
`Admin Alias`	The unique identifier for the Administration Server for the job(s).

Output

`rcadmin` shows the following information for each job:

Argument	Description
`Statistics on following job(s):`	The job's number. When you create a job, a number is assigned to it for reference purposes. The first job added is job #1, the second job added is job #2, and so on.
`Job Name:` *`jobname`*	The job's name.
`Job Host:` *`hostname`*	The unique identifier for the host where the job resides.
`Description`	A text description of this job.

Argument	Description

`Job Status`	The current running state of the job. Possible values are:
	`Unknown`
	`Created`
	`Inactive`
	`Running`
	`Paused`
	`Stopping`
	`Stopped`
	`Finished`
	`Scheduled`
	`Aborting`
	`Aborted`
	`Purging`
	`Deleting`
	`Interrupted`

`Collection Index Job:`	
`Number of docs crawled:`	The number of documents that were crawled by K2 Spider during the job.
`Number of docs inserted:`	The number of documents submitted to VDK for indexing during the job.
`Number of docs indexed:`	The number of documents indexed into the VDK collection during the job.
`Number of docs skipped: (due to duplication)`	The number of documents skipped during the job because they are duplicates of previously encountered documents.
`Number of docs skipped: (due to error message 404's)`	The number of documents skipped during the job because they do not exist, such as documents that result in an HTTP 404 error.
`Total num. of docs skipped:`	The total number of documents skipped during the job. For example, a document may be skipped because of the document's modification date, size, or MIME type.
`Number of docs deleted:`	The number of documents deleted from the VDK collection during the job.

Argument	Description
`Number of bad docs:`	The number of documents which VDK has trouble indexing and which it reports as "bad documents."
`Number of candidate docs:`	The number of documents waiting in the Controller to be sent to the Crawler.
`Number of docs being crawled:`	The number of documents the Controller has sent to the Crawler and the Crawler is currently processing.
`Total number of docs failed to fetch:`	The total number of documents retrieved from crawling.
`Number of child docs indexed:`	The number of child documents indexed into the VDK collection during the job. A *child* document is an attachment to a document that is crawled. Child documents most often come from the Lotus Notes gateway.
`Number of child docs deleted:`	The number of child documents deleted from the VDK collection during the job. A *child* document is an attachment to a document that is crawled. Child documents most often come from the Lotus Notes gateway
`Number of bad child docs:`	The number of child documents which VDK has trouble indexing and which it reports as "bad documents." A *child* document is an attachment to a document that is crawled. Child documents most often come from the Lotus Notes gateway
`Time in sec.since job started:`	The number of seconds since the job was started.
`Error Job(s):` *`jobname`*	The name of jobs that generated errors.

jobstart

The `jobstart` command enables you to run an existing K2 Spider indexing job.

After a job has run a log file is created in the directory *`dataDir`*`/jobs`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). It contains any text output that was generated by the command-line tool.

See the *Verity K2 Dashboard Administrator Guide* for more information on job logs.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Trigger chain action upon the job completion? [y|n]`	If you want to start this job and its subsequent chained jobs, type `y`. If you want to start only this job, without triggering its chained jobs, type `n`.
`Continue?`	Type `y` to start the job.

jobstop

The `jobstop` command enables you to stop an indexing job. The job stops after all normal processes execute.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Trigger chain action upon the job completion? [y|n]`	If you want to stop this job and its subsequent chained jobs, type `y`. If you want to stop only this job, without stopping its chained jobs, type `n`.
`Continue? [y|n]`	Type `y` to stop the job.

jobpause

The `jobpause` command enables you to pause an indexing job that is in progress. Use the `jobresume` command to restart the job. See "jobresume" on page 279.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Continue?`	Type `y` to pause the job.

jobresume

The `jobresume` command enables you to restart an indexing job after it has been paused using the `jobpause` command.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Continue?`	Type `y` to restart the job.

jobpurge

The `jobpurge` command enables you to purge the job that originally indexed the documents that now need removal from a 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.

This means all records in the collection(s) are not necessarily deleted. For example, say JobA indexes documents from the starting point `c:\projects` into collection1; and JobB indexes documents from the starting point `c:\setup` into the same collection. When you purge JobA, only the document records from `c:\projects` are deleted. Likewise, if collection1 and collection2 are indexed by JobC, and the job is purged, the records originating with JobC in both collections are deleted.

A job purge is useful when the properties of one job are changed, and multiple jobs are associated with the collection.

When style file changes are made that affect the schema, you must purge *all* document records from the collection using the `collpurge` command. See "collpurge" on page 217.

Note If a job indexed many documents, purging the job may take a significant length of time. To purge the job more quickly, delete the records from the collection(s) using the `collpurge` command, and then delete the records from the job itself using the `jobpurge` command.

Note Before purging a job, it is recommended the associated collection(s) be offline. This allows the collection to be *squeezed* after the purge. Squeezing a collection removes deleted documents from the collection, which recovers space and improves search performance. If the collection is not squeezed, the document records are marked as "deleted," but are not physically removed from the collection.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Continue?`	Type `y` to purge the collections in the job.

jobdel

The `jobdel` command enables you to remove an indexing job from the Administration Server. This command removes the job from the `admin`*N*`.xml` file, and from the directory *`dataDir`*`/jobs`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Continue?`	Type `y` to delete the job.

Note If this job triggers another job, as part of a chain of jobs, you are warned that deleting the job will break the chain, and asked whether you want to continue. If you delete the job, any jobs that are started by the deleted job will not run. You may want to create a new job chain, or delete the chained jobs first.

jobaddindex

The `jobaddindex` command enables you to add a collection to an indexing job. It specifies the collections that are populated with documents when you run the indexing job.

Note The gateway used by the collection must be the same as the gateway specified in the job.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Number of collections to add:`	The number of collections you want to add to this job.
`Collection #`*`n`*	
`Collection Alias`	The unique identifier for the first collection.
`Admin Alias`	The unique identifier for the Administration Server with which this collection is registered.

jobremoveindex

The `jobremoveindex` command enables you to delete a collection from an indexing job. This only removes the entries in the `admin`*N*`.xml` file. It does not delete any files.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The job's name.
`Admin Alias`	The unique identifier for the Administration Server for this job.
`Number of collections to remove:`	The number of collections you want to remove from this job. To remove all collections associated with this job, press the `Enter` key.
`Collection #`*n*	The number of the collection you want to remove.
`Collection Alias`	The unique identifier for the collection.
`Admin Alias`	The unique identifier for the Administration Server with which this collection is registered.

9

Managing User Defined Jobs

This chapter describes the commands used to manage user defined jobs.

- Introduction
- Managing User Defined Jobs
- Controlling User Defined Jobs
- Managing Custom Job Types

Introduction

The commands in this section manage *user defined jobs*. User defined jobs specify details for a task. However, unlike K2 Spider indexing jobs, the task is not limited to the indexing process. In fact, user defined jobs give you the flexibility to set up tasks with any command-line tool that is defined in the `verity.cfg` directly from `rcadmin`. These can be Verity tools such as `mkvdk`, `mktopics`, `mkpi`, or other utilities, such as `.cmd` and `.bat` files in Windows or shell scripts in UNIX. See the *Verity K2 Dashboard Administrator Guide* for information on defining commands in the `verity.cfg` file.

Following are examples of tasks you can accomplish with a user defined job:

- optimize a collection
- update a topic set
- update a parametric index

User defined jobs allow you to automate administration: you can run a user defined job right away, or schedule it to run later. You can also "chain" jobs. This means you set up two or more jobs to run in sequence, creating a "chain reaction."

User defined jobs are stored in the directory *`dataDir`*`/jobs`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). A job log named `console.log` is created when you run a job for the first time, and contains any text output that was generated by the command-line tool. It is also stored in the `jobs` directory.

Creating a User Defined Job

To create a user defined job, follow these steps:

1. Use the `udjcreate` command to create the job. See "udjcreate" on page 289. To create a job that uses a custom job type use the `udjset` command. See "Custom Job Types" on page 287, and "udjsset" on page 301.

   These commands enable you to specify a command-line tool and its associated arguments, schedule the job and create a job chain for the job. See "Creating a Job Chain" on page 287 for more information.

2. If you did not set up a schedule to run the job when you created the job, use the `jobstart` command to run the job right away, or use the `udjmodify` command to set up a schedule. See "udjstart" on page 299 and "udjmodify" on page 294.

Custom Job Types

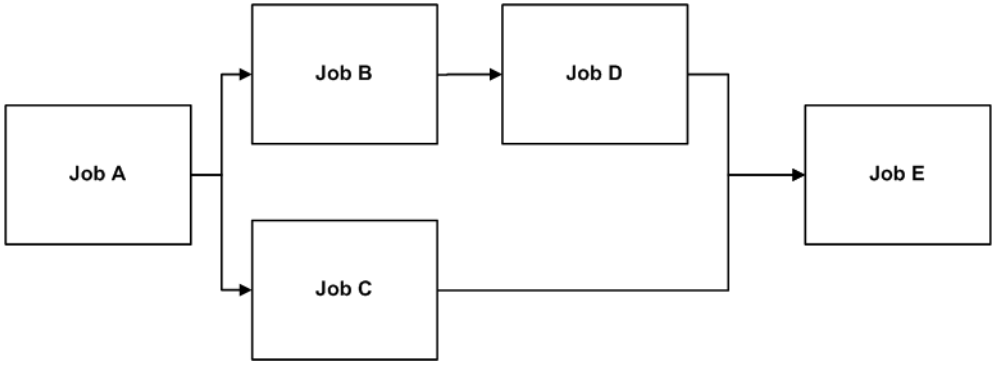
You can create user defined jobs that use custom job types. A custom job type has pre-defined, custom options that are used to build the command arguments for a job. The values for the custom options are stored in the job's `User data` field. When you create a job with a custom job type in K2 Dashboard, the values in the job type's user data are used to populate a secondary dialog that is accessed from the Create/Edit Job Properties dialog. See the `udjsset` command for more information custom job types.

Creating a Job Chain

You can create a *chain* of indexing jobs using the `udjcreate` or `udjmodify` commands. A chain reaction occurs when you link one job—either a a K2 Spider indexing job or a user defined job—to another. Each chained job can have one or more prior jobs. When any prior job runs to completion, the job is started. This allows you to set up sequences of jobs. You are not limited in the amount of jobs you chain together.

Figure 9-1 shows an example of jobs and predecessors for each.

Figure 9-1 Multiple Chained Predecessors



In this example, the following sequence occurs:

1. Job A runs and finishes.
2. Jobs B and C are triggered and start.
3. When Job B finishes, Job D is triggered and starts.
4. When Job C or D finishes, Job E is triggered and starts.

Notice in step 4 that only one of Job C or D must finish before Job E is triggered to start. In this case, when one of the prior jobs completes, it triggers Job E to start. If the other prior job finishes while Job E is still in a `Running` state, its trigger is queued, as Job E is already underway. When Job E completes, the queued trigger runs Job E a second time. Likewise, if Job E had finished before its second triggering, it would be started again immediately.

Managing User Defined Jobs

The commands in this section enable you to create, modify, and delete user defined jobs.

This section describes the following commands:

- udjcreate
- udjmodify
- udjget
- udjdel

udjcreate

The `udjcreate` command enables you to create and schedule generic user defined jobs. With this command, you can also create a job chain.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The case-sensitive name of the user defined job. The job alias must be unique to the host.
`Admin Alias`	The alias of the Administration Server for this job.
`Description`	A text description of this job. This is optional.
`Command`	The command that performs the task associated with the job. For example, `mkvdk`, `mkre`, `mkpi`. This command is run with the arguments defined in the `Arguments` prompt. You can run a Verity command or a third-party command that is defined in the `verity.cfg`. See the *Verity K2 Dashboard Administrator Guide* for information on defining commands in the `verity.cfg`.

Argument	Description

`Argument`	The argument(s) for the command. This is optional. For example, typing `mkvdk` at the `Command` prompt, and typing
	`-create -collection my_collection_path`
	at the `Arguments` prompt is the same as typing the command and arguments on the command-line:
	`mkvdk -create -collection my_collection_path`
	This creates a new, empty collection.
	You can use tokens in the command arguments to specify paths or Verity locales. For more information, see "Using Tokens in Command Arguments" on page 292.

`Start Chained Job only on Success?`	If you want this job to run only when a prior chained job succeeds, set this to `y`. If you want this job to run when a prior job completes, even if that job fails or aborts, set this to `n`. This is useful for jobs that perform cleanup procedures. If you set up multiple prior jobs, and then enable this setting, at least one of the prior jobs must succeed before this job can run.
`Associate an Index?`	Indicates whether an index is associated with this job. If the job is associated with an index, then you cannot delete the index without first disassociating the index (using the `udjmodify` command), or deleting the job. To view the indexes associated with a job, use the `hierarchyview` command. See "hierarchyview" on page 38.
`Index Type`	The type of index that you want to associate with the job. The following options are available: `c` = collection. `t` = knowledge tree. `p` = parametric index. `s` = topic set. `r` = recommendation index.
`Index Alias`	The alias of the index that you want to associate with the job.
`Create job chain?`	If you want the completion of another job to trigger the start of this job, type `y`. To view the relationship a job has to other jobs in the chain, use the `udjget` command. Each chained job must already exist; and all jobs in a chain must be on the same host.
`Number of chained jobs`	The number of jobs that can trigger the start of this job.

Argument	Description
`Chained Job #`	This section identifies each job that runs prior to this job.
`Job Name`	The name of the job that runs prior to this job.
`Schedule this job?`	Type `y` to create a schedule for the job you are creating. You can create a schedule for a job that is already part of a job chain. If the job is already running because it was started by a predecessor job, then the scheduled start is ignored. Before you schedule a parametric index job, ensure the job does not include the argument `-persist`.
`Frequency`	How frequently the job runs. The available options are: `o` = Runs the job a single time according to the values set in the `Year/Month/Day/Hour/Minute` prompts. `h` = Runs the job hourly according to the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `d` = Runs the job every day at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `w` = Runs the job weekly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `k`= Runs the job on weekdays only, at the time indicated in the `Year/Month/Day/Hour/Minute` prompts `m` = Runs the job monthly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. If you select this option, and the `Day` is `31`, indicating "Monthly on the 31st," the job still runs during months with less than 31 days. In these cases, the job runs on the last day of the month, such as February 28th. `e` = Runs the job monthly, on a particular weekday, such as the second Monday of each month. `i` = Runs the job periodically at specified intervals (minutes).
`Repeat interval (mins)`	If `Frequency` is `i`, you see this prompt. Enter the interval in minutes at which the job runs. For example, if you set the repeat interval to 300, the job runs every 5 hours. The maximum interval is 32 days (32*24*60-1 = 46079 minutes).
`Start the job now?`	To start the job immediately, type `y`. To start the job at a specified date and time, type `n`. If you type `n`, you see the following prompts:

Argument	Description
`Year`	The year that the job starts to run (`yyyy`).
`Month`	The month that the job starts to run (1-12).
`Day`	If `Frequency` is anything other than `e`, you see this prompt. Enter the day that the job starts to run (1-31).
`Weekday [(M)onday|(T)uesday|(W)ednesday|T(h)ursday|(F)riday|(S)aturday|S(u)nday]`	If `Frequency` is `e`, you see this prompt. Choose the day of the week that the job runs.
`Week of Month [(1)st|(2)nd|(3)rd|(4)th|(l)ast]`	If `Frequency` is `e`, you see this prompt. Choose the week in the month that the job runs.
`Hour`	The hour the job starts to run. (0-23)
`Minute`	The minute the job starts to run. (0-59)

Using Tokens in Command Arguments

You can use the following tokens in the command arguments to specify paths or locales:

`$GETPATH(COLL,`*`alias`*`)$` This token is replaced with the path to the collection with the specified alias. It is only valid if the collection is registered on the host.

`$GETPATH(KTREE,`*`alias`*`)$` This token is replaced with the path to the knowledge tree with the specified alias. It is only valid if the knowledge tree is registered on the host.

`$GETPATH(PI,`*`alias`*`)$` This token is replaced with the path to the parametric index with the specified alias. It is only valid if the parametric index is registered on the host.

`$GETPATH(CFG,`*`alias`*`)$` This token is replaced with a path that is defined in the `[mapping]` section of the `verity.cfg` file with the specified alias. The `data` alias is useful, as it retrieves the K2 data path.

`$GETPATH(JOB,alias)$` This token is replaced with the path to the job with the specified alias.

`$GETLOCALE(`*`type`*`,`*`alias`*`)$` This token is replaced with the Verity locale of the collection or knowledge tree. In this case, `type` indicates `COLL` or `KTREE`.

For example, the following command, arguments, and tokens:

```
mkpi -collPath $GETPATH(COLL,mycoll)$ -piPath $GETPATH(PI,mypi)$
```

could translate to the following, depending on configuration:

```
mkpi -collPath c:\verity\data\colls\mycoll -piPath c:\verity\
data\pis\mypi
```

NOTES

- If `rcadmin` displays the message "The command is not allowed to run. The command name must be specified in the `verity.cfg` file. (-6171)," the command is not defined in the `verity.cfg` file. For information on correcting this, see "Defining Commands in the verity.cfg File" in the *Verity K2 Dashboard Administrator Guide*.

- If `rcadmin` displays the message "Failed to launch the command specified in the user defined job. (-6206)," the related executable is not stored in the proper directory. If you do not store the command in the `bin` directory, you must enter a fully qualified path at the `Command` prompt when you create the user defined job. For example, rather than simply entering the command name, such as `myprogram`, you would enter the absolute path

  ```
  c:\program files\myprogram
  ```

  In Windows, you must qualify BAT or CMD scripts with a `.bat` or `.cmd` extension.

  For non-standard script languages such as Perl, you would run the Perl interpreter as the command and make the script the first argument, as follows:

  ```
  c:\perl\bin\perl c:\myperlscript\mkpi.pl
  ```

udjmodify

The `udjmodify` command enables you to modify the parameters for an existing user defined job.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The case-sensitive name of the user defined job.
`Admin Alias`	The alias of the Administration Server for this job.
`Description`	A text description of this job.
`Command`	The command that performs the task associated with the job.
`Argument`	The arguments you want to run with the command. This is optional.
`Start Chained Job only on Success?`	If you want this job to run only when a prior chained job succeeds, set this to y. If you want this job to run when a prior job completes, even if that job fails or aborts, set this to n. This is useful for jobs that perform cleanup procedures. If you set up multiple prior jobs, and then enable this setting, at least one of the prior jobs must succeed before this job can run.
`Associate an Index?`	If an index is not currently associated with the job, you see this prompt. Type `y` to associate an index. You are prompted for the index type, and index alias. See "udjcreate" on page 289 for information on associating an index.
`Index associated:`	If an index is already associated with the job, the index type, index alias and host alias are displayed.
`Index type`	The type of index associated with the job (collection, knowledge tree, parametric index, topic set, or recommendation index). (Read only.)
`Index Alias`	The alias of the index associated with the job. (Read only.)
`Host Alias`	The alias of the host for this index. (Read only.)

Argument	Description
`(r)emove, (c)hange Associated Index, or (n)o action?`	If an index is already associated with the job, you see this prompt. To remove the association with the current index, type `r`. To change the associated index, type `c`. To exit this prompt with no changes, type `n`. If you change the associated index, you are prompted for the index type and alias of the new index.
`Create job chain?`	If a job chain is not defined for this job, you see this prompt. Type `y` to define a job chain. You are prompted for the number of jobs that trigger this job, and the name of each job. See "udjcreate" on page 289 for information on creating a job chain.
`Number of chained jobs:`*`n`*	If a job chain is already defined for this job, the number of jobs that triggers the start of this job is displayed. (Read only.)
`Job #`*`n`*`:` *`JobName, AdminAlias`*	This section identifies each job that triggers the start of this job. It displays the job name and the alias of the Administration Server for the job. (Read only.)
`(r)emove, (c)hange job chain, or (n)o Action?`	If a job chain is already defined for this job, you see this prompt. To remove the job chain, type `r`. To change the job chain, type `c`. To exit this prompt with no changes, type `n`. If you are changing the job chain, you are then prompted for the number of chained jobs and the name(s) of the chained jobs.
`Schedule this Job?`	If this job is not currently scheduled, you see this prompt. Type `y` to create a schedule for this job. See the `udjcreate` command for more information on creating a schedule.
`Scheduling information`	If a schedule is already defined for this job, the mode and frequency of the schedule are displayed.
`Mode`	Indicates whether the schedule is enabled or disabled. (Read only.)
`Frequency`	The scheduled job execution frequency. (Read only.)
`(d)isable/(e)nable, (c)hange job schedule, or (n)o Action taken?`	To disable an enabled schedule, type `d`. To enable a disabled schedule, type `e`. To change the schedule, type `c`. To exit the prompt with no changes, type `n`.

udjget

The `udjget` command enables you to view information about generic user defined jobs.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the user defined job. To get information on all user defined jobs on this host, do not enter a name at this prompt.
`Admin Alias`	The alias of the Administration Server for this job.

Output

`rcadmin` shows the following information for each user defined job:

Argument	Description
`Job #`*n*	The number of the user defined job.
`Job Name`	The name of the user defined job.
`Host Alias`	The alias of the host where the job resides.

`Status`	The current state of the job. Possible values are as follows:
	`Created`
	`Running`
	`Finished`
	`Scheduled`
	`Deleting`
	`Stopping`
	`Stopped`
	`Aborted`
	`Interrupted` (This state displays if the job did not finish successfully.)

`Description`	A text description of this job.
`Command`	The command that runs in this job.
`Argument`	The arguments that run with the command.

Argument	Description
`Scheduling information:`	
`Mode`	Indicates whether the schedule is enabled or disabled.
`Frequency`	The scheduled job execution frequency.
`Index associated:`	If an index is associated with the job, the index type, index alias and host alias are listed.
`Index type`	The type of index associated with the job (collection, knowledge tree, parametric index, topic set, recommendation index.
`Index Alias`	The alias of the index associated with the job.
`Host Alias`	The alias of the host for this index.
`Start Chained Job only on Success`	Indicates whether this job runs only when a prior chained job succeeds; or whether this job runs even if a prior chained job fails or aborts.
`Chained Jobs that run before this job:`	The number of jobs that are run before this job starts.
`Job #`*`n`*`:` *`jobname`*	The name of the job.
`Chained Jobs that run after this job:`	The number of jobs that are started after the completion of this job.
`Job #`*`n`*`:` *`jobname`*	The name of the job.

udjdel

The `udjdel` command removes a user defined job from the Administration Server. This command removes the job from the `admin`*N*`.xml` file, and from the directory *`dataDir`*`/jobs`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX).

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the user defined job.
`Admin Alias`	The alias of the Administration Server for this job.

Note If this job is part of a chain of jobs, you are warned that deleting the job will break the chain, and asked whether you want to continue. If you delete the job, any jobs that are started by the deleted job will not run. You may want to create a new job chain, or delete the chained jobs first.

Controlling User Defined Jobs

The commands in this section enable you to start, stop, and abort user defined jobs.

This section describes the following commands:

- udjstart
- udjstop
- udjabort

udjstart

The `udjstart` command enables you to start a user defined job. After a job has run a log file is created in the directory *`dataDir`*`/jobs`, where *`dataDir`* is the pathname of the installation's data directory (for example, `usr/verity/data` on UNIX). It contains any text output that was generated by the command-line tool.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the user defined job.
`Admin Alias`	The alias of the Administration Server for this job.
`Trigger chain action upon the job completion?`	If you want to start this job and its subsequent chained jobs, type `y`. If you want to start only this job, without triggering its chained jobs, type `n`.
`Continue?`	Type `y` to start the job.

udjstop

The `udjstop` command enables you to stop a user defined job. The job stops after all normal processes execute.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the user defined job.
`Admin Alias`	The alias of the Administration Server for this job.
`Continue?`	Type `y` to stop the job.

udjabort

The `udjabort` command enables you to abort a user defined job. The job ceases immediately, and not all processes complete.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The name of the user defined job.
`Admin Alias`	The alias of the Administration Server for this job.
`Continue?`	Type `y` to abort the job.

Managing Custom Job Types

The command in this section manage user defined jobs with custom job types.

This section describes the following commands:

- udjsset
- udjsget

udjsset

The `udjsset` command enables you create or modify a user defined job with a custom job type. A custom job type has pre-defined, custom options that are used to build the command arguments for a job. The values for the custom options are stored in the job's `User data` field. When you create a job with a custom job type in K2 Dashboard, the values in the job type's user data are used to populate a secondary dialog that is accessed from the Create/Edit Job Properties dialog.

There are four custom job types that ship with K2:

- `Collection_Swap`
- `Optimize_Collection`
- `Update_Parametric_Index`
- `Update_Recommendation_Indexes`

You can use these types as is, you can enhance them to suit your particular needs, or you can create entirely new custom job types. For detailed procedures on creating user defined jobs with these two shipped job types, and on creating a new custom job type, see the *Verity K2 Dashboard Administrator Guide*.

Input

Enter the following information when prompted:

Argument	Description
`Job Name`	The case-sensitive name of the user defined job. The job alias must be unique to the host.
`Admin Alias`	The alias of the Administration Server for this job.
`Modify Type`	Indicates whether to modify an existing user defined job or create a new one. The following options are available: `0` = Update an existing user defined job. `1` = Insert a new user defined job.
`Job Type`	The name of the custom job type. This is the unique portion of the name of the HTML file that was created for this custom job type. The HTML file follows the naming convention `udj`*`JobTypeName`*`.html`, where *`JobTypeName`* is the unique name of the job type. The HTML files for custom job types are stored in the directory *`installDir`*`/k2_6/appserver/webapps/verity_dashboard/udj`, where *`installDir`* is the pathname of the Verity installation directory (for example, `usr/verity/` on UNIX, or `C:\Program Files\Verity` on Windows). The *`JobTypeName`* in `udj`*`JobTypeName`*`.html` is displayed in the K2 Dashboard's drop-down list in the **Job Type** field of the Create/Edit Job Properties dialog. You should not modify the job type after a job is created.

Argument	Description
`User data`	The custom fields and values that are required for this job type. These fields and values are used to populate the secondary dialog that is accessed from the Create/Edit Job Properties dialog in K2 Dashboard. You can modify the value of any "field/value" pair. To determine what user data is defined for a job type, use the `udjsget` command. For example, the `Update_Recommendation_Indexes` type contains the following user data: `IndexOptions=update;;verbose=1` The first field, `IndexOptions`, corresponds to the **Common Index Management Options** in the secondary dialog for recommendation index jobs. Its value is set to `update` all indexes. The second field, `verbose`, corresponds to the **Status Log Level** field in the secondary dialog for recommendation index jobs. Its value is set to `1`, which is `Trace Level 1`. The format of the user data is determined by what method is used to create the custom job type interface. For the custom job types that are shipped with K2, if an invalid field is entered, the field is ignored, and the default is used.
`Description`	A text description of this job. This is optional.
`Command`	The command that performs the task associated with the job. For example, `mkvdk`, `mkre`, `mkpi`. This command is run with the arguments defined in the `Arguments` prompt. You can run a Verity command or a third-party command that is defined in the `verity.cfg`. See the *Verity K2 Dashboard Administrator Guide* for information on defining commands in the `verity.cfg`.

`Argument`	The argument(s) for the command. This is optional. For example, typing `mkvdk` at the `Command` prompt, and typing
	
	`-create -collection my_collection_path`
	
	at the `Arguments` prompt is the same as typing the command and arguments on the command-line:
	
	`mkvdk -create -collection my_collection_path`
	
	This creates a new, empty collection.
	
	You can use tokens in the command arguments to specify paths or Verity locales. For more information, see "Using Tokens in Command Arguments" on page 292.

Argument	Description
`Trigger Chained Job only on Success?`	If you want this job to run only when a prior chained job succeeds, set this to `y`. If you want this job to run when a prior job completes, even if that job fails or aborts, set this to `n`. This is useful for jobs that perform cleanup procedures. If you set up multiple prior jobs, and then enable this setting, at least one of the prior jobs must succeed before this job can run.
`Associate an Index?`	Indicates whether an index is associated with this job. If the job is associated with an index, then you can not delete the index without first disassociating the index (using the `udjmodify` command), or deleting the job. To view the indexes associated with a job, use the `hierarchyview` command. See "hierarchyview" on page 38.
`Index Type`	The type of index that you want to associate with the job. The following options are available: `c` = collection. `t` = knowledge tree. `p` = parametric index. `s` = topic set. `r` = recommendation index.
`Index Alias`	The alias of the index that you want to associate with the job.
`Create job chain?`	If you want the completion of another job to trigger the start of this job, type `y`. To view the relationship a job has to other jobs in the chain, use the `udjget` command. See "udjget" on page 296. Each chained job must already exist; and all jobs in a chain must be on the same host.
`Number of chained jobs`	The number of jobs that can trigger the start of this job.
`Chained Job #`	This section identifies each job that runs prior to this job.
`Job Name`	The name of the job that runs prior to this job.

Argument	Description
`Schedule this job?`	Type `y` to create a schedule for the job you are creating. You can create a schedule for a job that is already part of a job chain. If the job is already running because it was started by a predecessor job, then the scheduled start is ignored. Before you schedule a parametric index job, ensure the job does not include the argument `-persist`.
`Frequency`	How frequently the job runs. The available options are: `o` = Runs the job a single time according to the values set in the `Year/Month/Day/Hour/Minute` prompts. `h` = Runs the job hourly according to the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `d` = Runs the job every day at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `w` = Runs the job weekly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. `k`= Runs the job on weekdays only, at the time indicated in the `Year/Month/Day/Hour/Minute` prompts `m` = Runs the job monthly at the time indicated in the `Year/Month/Day/Hour/Minute` prompts. If you select this option, and the `Day` is `31`, indicating "Monthly on the 31st," the job still runs during months with less than 31 days. In these cases, the job runs on the last day of the month, such as February 28th. `e` = Runs the job monthly, on a particular weekday, such as the second Monday of each month. `i` = Runs the job periodically at specified intervals.
`Repeat interval (mins)`	If `Frequency` is `i`, you see this prompt. Enter the interval in minutes at which the job runs. The maximum interval is 32 days (32*24*60-1 = 46079 minutes).
`Start the job now?`	To start the job immediately, type `y`. To start the job at a specified date and time, type `n`. If you type `n`, you see the following prompts:
`Year`	The year that the job starts to run (`yyyy`).
`Month`	The month that the job starts to run (1-12).

Argument	Description
`Day`	If `Frequency` is anything other than `e`, you see this prompt. Enter the day that the job starts to run (1-31).
`Weekday [(M)onday|(T)uesday|(W)ednesday|T(h)ursday|(F)riday|(S)aturday|S(u)nday]`	If `Frequency` is `e`, you see this prompt. Choose the day of the week that the job runs.
`Week of Month [(1)st|(2)nd|(3)rd|(4)th|(l)ast]`	If `Frequency` is `e`, you see this prompt. Choose the week in the month that the job runs.
`Hour`	The hour the job starts to run. (0-23)
`Minute`	The minute the job starts to run. (0-59)

udjsget

The `udjsget` command enables you to view information about user defined jobs that use custom job types. For detailed procedures on creating new custom job types, see the *Verity K2 Dashboard Administrator Guide*.

Input

Enter the following information when prompted:

Argument	Description
`Job Name (optional)`	The name of the user defined job. To get information on all user defined jobs on this host, do not enter a name at this prompt.
`Admin Alias`	The alias of the Administration Server for this job.

Output

`rcadmin` shows the following information for each user defined job:

Argument	Description
`Job #n`	The number of the user defined job.
`Job Name`	The name of the user defined job.
`Host Alias`	The alias of the host where the job resides.

`Status`	The current state of the job. Possible values are as follows:
	`Created`
	`Running`
	`Finished`
	`Scheduled`
	`Deleting`
	`Stopping`
	`Stopped`
	`Aborted`
	`Interrupted`

`Description`	A text description of this job.
`Command`	The command that runs in this job.
`Argument`	The arguments that run with the command.
`Job Type`	The name of the custom job type. This is the unique portion of the name of the HTML file that was created for this custom job type.

Argument	Description
`User data`	The custom fields and values that are required for this job type. These fields and values are used to populate the secondary dialog that is accessed from the Create/Edit Job Properties dialog in K2 Dashboard. For example, the `Update_Recommendation_Indexes` type contains the following user data: `IndexOptions=update;;verbose=1` The first field, `IndexOptions`, corresponds to the **Common Index Management Options** in the secondary dialog for recommendation index jobs. Its value is set to `update` all indexes. The second field, `verbose`, corresponds to the **Status Log Level** field in the secondary dialog for recommendation index jobs. Its value is set to `1`, which is `Trace Level 1`. The format of the user data is determined by what method is used to create the custom job type interface.
`Scheduling information:`	
`Mode`	Indicates whether the schedule is enabled or disabled.
`Frequency`	The scheduled job execution frequency.
`Index associated:`	If an index is associated with the job, the index type, index alias and host alias are listed.
`Index type`	The type of index associated with the job (collection, knowledge tree, parametric index, topic set, recommendation index.
`Index Alias`	The alias of the index associated with the job.
`Host Alias`	The alias of the host for this index.
`Start Chained Job only on Success`	Indicates whether this job runs only when a prior chained job succeeds; or whether this job runs even if a prior chained job fails or aborts.

Argument	Description
`Chained Jobs that run before this job:`	The number of jobs that are run before this job starts.
`Job #`*`n`*`:` *`jobname`*	The name of the job.
`Chained Jobs that run after this job:`	The number of jobs that are started after the completion of this job.
`Job #`*`n`*`:` *`jobname`*	The name of the job.

Index

Symbols

? command 25
@controller 119, 122, 126
@local 22

A

admincred.dat 24
adminget command 56
Administration Server 18, 54, 56
  configuring 54
  launching 20, 21
  starting 20, 21
  starting on UNIX 21
administration XML file
  saving a backup 19
administrator 23
adminkilllaunched command 118
adminlaunch command 116
admin*N*.xml 19, 127, 140, 144, 158, 263, 281, 298
  determining active 19, 57
adminset command 54
adminsignal command 117
attaching an index 194
authentication 23, 168
  LDAP 173, 178
  UNIX 178
  user-defined 175, 180
  Windows 170, 176
automating tasks 27

B

broker
  See K2 Broker
brokerattachget command 110
brokerattachlistget command 109
brokerattachset command 107
brokerget command 61
brokerset command 59

C

cache
  document object 200, 213, 223
  HTML 200, 213, 223
  search 62, 71, 131
  source query 71
character set 22, 71, 74, 173, 178, 199, 213, 222
-charmap 22
charmap 22, 71, 74, 173, 178, 199, 213, 222
checkuser command 23
collcreate command 209, 264
collection 205– 213
  adding to a job 282
  adding to a K2 system 205
  attaching 194
  creating 209
  deleting 192
  deleting from a job 283
  detaching 193
  indexing 264
  registering 206
  setting the locale for 199
  swapping 24
  viewing registration information 211
  viewing status 214, 232
collget command 211
collpurge command 217
collset command 206
collstatget command 214
CPU binding 48

custom job type 301– 309
Collection_Swap 301
Optimize_Collection 301
Update_Parametric_Index 301
Update_Recommendation_Indexes 236, 237, 245, 250, 301

D

deleting services 127
detaching services 105
document object cache 200, 213, 223
domain
See K2 domain
domain controller 171, 177
dynamic memory storage 182, 185

E

email notification 57
notification level 55
setting 54, 55
encrypted topic set 142
error message 25
errormessage command 25
exit command 27
exiting rcadmin 27
extbrokerattachset command 91
extbrokerdetach command 99
external services 90– 103
extsvcticketattachset command 97
extsvcticketdetach command 102
extticketattachset command 94
extticketdetach command 101

F

full restart 26, 119

G

getgroup command 23

H

help 25
help command 25
hierarchyview command 38
history factor 241, 251, 253
hostdomainget command 50
hostdomainset command 49
hostview command 39
HTML cache 200, 213, 223

I

index
attaching 194
changing state 197
configuring VDK settings 198
configuring viewing service settings 200
deleting 192
detaching 193
index server
See K2 Index Server
indexattach command 194
indexdel command 192
indexdetach command 193
indexerget command 88
indexerset command 86
indexing a collection 264
indexing job
adding a collection 282
deleting 281
deleting a collection 283
job chain 265, 267
modifying 271
pausing 279
purging collection in 280
restarting 279
running state of 276
stopping 278
indexserverget command 195
indexstateset command 197

`indexvdkset` command 74, 198
`indexview` command 39
`indexviewset` command 200

J

job chain 265, 267
`jobaddindex` command 205, 264, 282
`jobcreate` command 205, 264, 267
`jobdel` command 281
`jobgetdetail` command 273
`jobgetstats` command 275
`jobmodify` command 271
`jobpause` command 279
`jobpurge` command 280
`jobremoveindex` command 283
`jobresume` command 279
`jobstart` command 264, 278
`jobstop` command 278

K

K2 Administration
    starting 21
    stopping 21
K2 Broker 18, 58– 63, 107, 109, 110, 131
    configuring 59
K2 domain 18, 49, 50, 90, 110, 114
K2 Index Server 82– 89
K2 reporting 64
K2 Server 18, 64– 74, 107, 110, 131, 198
    configuring 65
    maximum active collection requests 67
    starting manually 21
K2 Spider 260– 263
    deleting registration information 263
    indexing job 264– 283
    registering 261
    viewing registration information 262
K2 Ticket Server 18, 79, 111, 112, 113
    configuring 76
`k2adminstart.sh` 21
`k2adminstop.sh` 21
knowledge base 70, 73, 199, 212, 222
knowledge tree 218– 223
    attaching 194
    deleting 192
    detaching 193
    registering 219
    setting locale for 199, 219
    viewing registration information 221

L

`label` command 19
launching commands via rcadmin 116
launching `rcadmin` 22
LDAP authentication 173, 178
LDAP storage 183, 186
license keys 42
    set 41
    update 47
`licensecheck` command 45
`licenseget` command 42
licenses
    managing 40– 48
`licenseservice` command 47
`licenseset` command 41
`licenseupdate` command 47
locale 71, 222
    setting for a collection 199
    setting for a knowledge tree 199, 219
    setting for a Profile Service 74, 136
`-log` 27
log
    configuring query 152
    configuring status 149
    configuring transaction 129
    query 147– 161
    status 148– 151
    transaction 63, 72

log file 27
    password in 28
logging in 23
logging in automatically 24
`login` command 23
login credentials 24
login modules 168, 170, 176
`logout` command 23

M

manual start
    Administration Server 20, 21
    K2 Server 21
mapterms
    See profile decryption
Master Administration Server 18, 22
maximum search time 60, 62, 67, 69, 208, 212, 220, 222
mirrored collection 60, 62
`mkpi` 229, 286, 289, 303
`mkre` 239, 242, 289, 303
`mktopics` 286
`mkvdk` 286, 289, 303
multi K2 domain support
    See K2 domain

N

notification email address 57
    notification level 55
    setting 54, 55

P

`paraget` command 231
parametric index 228– 231
    attaching 194
    deleting 192
    detaching 193
    registering 229
`paraset` command 229
`parastatget` command 232
persistent store module 80, 168, 181– 184
ping delay 60, 62, 66, 69
pre-authentication 81
    and security 78
    enabling 77
process ID 121
profile decryption 235, 237, 239, 240
Profile Net 133, 143– 145
    configuring 141
    deleting 144
    setting state 145
Profile Service 133– 140
    configuring 135
    deleting 140
    setting locale for 74, 136
`profiledel` command 140
`profileget` command 138
`profilenetdel` command 144
`profilenetget` command 143
`profilenetset` command 141
`profilenetstateset` command 145
`profileset` command 135
purge
    a collection 217
    an indexing job 280

Q

`q` command 27
query log 147– 161
    configuring 152
`querylogdel` command 158
`querylogexcludedel` command 160
`querylogexcludeget` 160
`querylogexcludeset` command 159
`querylogget` command 156
`querylogset` command 152
queue polling 60, 62, 67, 70
quick restart 26, 119
`quit` command 27

R

`rck2` 60, 62, 66, 69
Recommendation Engine
  configuring host-wide defaults 237
  getting host-wide defaults 240
recommendation index 234– 250
  attaching 194
  deleting source information 249
  detaching 193
  getting host-wide defaults 240
  history factor 241, 251, 253
  registering 242
  registering a data source 245
  score factor 238, 240, 241, 251, 253
  setting host-wide defaults 237
  viewing data source registration information 248
  viewing registration information 244
recommendation index type 250– 257
  deleting relevance information for 257
  relevance information 241, 253, 256
`Refresh_Recommendation_job` 236, 237, 245, 250
`reget` command 240
registering indexes with an Administration Server
  collection 206
  knowledge tree 219
  parametric index 229
  recommendation index 242
  topic set 225
reporting 64
`reset` command 237
restart
  full 26, 119
  quick 26, 119
results cache
  See search cache
`retxlogset` command 129
`riget` command 244
`riset` command 242
`risourcedel` command 249
`risourceget` command 248
`risourceset` command 245
`ritypedel` command 253
`ritypeget` command 252
`rityperelde1` command 257
`rityperelget` command 256
`rityperelset` command 254
`ritypeset` command 250

S

score factor 238, 240, 241, 251, 253
scripts
  creating 27
  creating log file 27
  maintaining 28
  upgrading 28
search cache 62, 71, 131
search time
  maximum 60, 62, 67, 69, 208, 212, 220, 222
`searchcacheset` command 131
server
  See K2 Server
`servercpuset` command 48
`serverget` command 68
`serverset` command 65
`servicecheckstate` command 125
`servicedel` command 127
`servicedetach` command 105
`servicegetpid` command 121
services
  deleting 127
  detaching 105
  process ID 121
  starting and stopping 115– 124
  viewing configured state of 125
  viewing running state of 126
  watched 121, 122, 123
`servicesignal` command 26, 119

servicestartup command 120
servicestateget command 126
servicewatchedstate command 121
shutdown 119
source query cache 71
spider
    See K2 Spider
spiderdel command 263
spiderget command 262
spiderset command 261
starting and stopping services 115– 124
starting K2 Administration
    on UNIX 21
starting rcadmin 22
state
    configured 61, 68, 79, 196
    running 276
    viewing configured service state 125
    viewing running service state 126
status log 148– 151
    configuring 149
statuslogget command 151
statuslogset command 149
stopping K2 Administration
    on UNIX 21
styledel command 204
styleget command 137, 202, 207, 209
styleset
    deleting 204
    registering 201
    viewing 137, 202, 207, 209
styleset command 201
swapping collections 24

T

ticket server
    See K2 Ticket Server
ticketattachget command 113
ticketattachlistget command 112
ticketattachset command 111
ticketget command 79
ticketlmget command 176
ticketlmset command 170
ticketpsmget command 185
ticketpsmset command 182
ticketset command 76
token
    in command argument 292
topic set 70, 73, 83, 85, 87, 136, 138, 192, 199, 212, 222, 224– 226
    attaching to a collection 227
    detaching from a collection 227
    encrypted 142
    registering 225
topic sets 224
topics 224
topicsetcollattach command 227
topicsetcolldetach command 227
topicsetget command 226
topicsetset command 225
transaction log 63, 72
    configuring 129
treeget command 221
treeset command 219
trusted client 164– 167
    adding 165
    deleting 167
trustedclientdel command 167
trustedclientget command 166
trustedclientset command 165

U

udjabort command 300
udjcreate command 289
udjdel command 298
udjget command 296
udjJobTypeName.html 302
udjmodify command 294
udjsget command 306
udjsset 301

`udjstart` command 299
`udjstop` command 300
UNIX authentication 178
user defined job 285– 309
    aborting 300
    and indexing jobs 265
    creating 289
    custom job type 301– 309
    deleting 298
    job chain 268, 281, 287, 289, 290, 298, 304
    modifying 294
    scheduling 268, 272, 291, 295, 305
    starting 299
    stopping 300
    token 290, 292, 303
    user data 287, 301, 308
`USER_PARAMS` key
    filtering from K2 78, 80
user-defined authentication 175, 180
user-defined storage 184, 187

V

valid user
    check 23
VDK settings 198
    configuring 73, 198
`vdksettingset` command 73
`VdkWeightedFeatures` 245, 246, 247, 249
Verity Control Module 21, 115
`verity.cfg` 20, 116
`version` command 28, 29
view
    K2 system arranged by host 39
    K2 system hierarchy 38
Viewing Service 200, 213, 223

W

watched services 121, 122, 123
Windows authentication 170, 176
`wsadd` command 122
`wsdel` command 123
`wsdelall` command 123
`wsget` command 123