COLLECTION BUILDING: FEATURES AND OPTIONS

COLLECTION BUILDING: FEATURES AND OPTIONS

Verity collections are indexed meta-data about a set of documents that have been processed so they are retrievable by Verity Products. Collections are created using Verity utility programs. In this section we will cover two of these indexers, the vspider utility and the mkvdk utility. These indexers are both very effective in locating the important attribute and search information through sophisticated formatting and filtering tools. During processing, indexes are populated and automatically optimized to enable fast searching of:

- Words contained in the documents
- Attribute information about the documents like their titles or authors

A collection is a series of indexes, which store data about your documents. Stored efficiently within each index, this data consists of:

- Location information
- Standard fields available for searching
- Custom fields you define for your documents
- All the words in each document and word location information to allow concept, Boolean and proximity searching.

THE COLLECTION DIRECTORY

The collection directory contains all the files that are needed to provide access to your documents or to manage the collection. The main job of the collection is to store information about the document (attributes) and to store the word index. Because you index at different times, or index a large amount of data, the collection is optimized into smaller components called partitions, each having these two indexes (attributes and words) for a group of documents in the collection. A collection can have many partitions. Depending upon how the collection was created, there are other directories in the collection that store data. Here is an overview of the basic role of these various collection directories. Remember that all collections do not have all of the same sub-directories, they only have the ones they need.

The **ASSISTS Directory** stores spanning word lists and information about the collection The **PARTS Directory** stores partitions, consisting of two key elements, the ".ddd" which holds the attribute information and the ".did" which holds the word index.

The .ddd file is a binary list of all of the documents in a partition. There is a separate record for each document, which includes several internal fields (like the document name) along with user-defined fields. The .did file is a binary database of all the words in all of the documents in a partition. If the documents are large and the partition is large, the .did files will be quite large as well. The general rule of thumb is that the .did will probably be about 40% of the size of the original documents.

The **PDD Directory** stores the partition index, which manages all the partitions in the collection. The **STYLE Directory** stores configuration files. The **TRANS Directory** stores transaction files used in auto-administration. Other collection directories you may find, are the **TOPICIDX Directory** which stores topic indexes for each partition, **TDE Directory** which stores work files for Topic Document Entry, **WORK Directory** which stores task files used in auto-administration, **MORGUE Directory** which stores outdated files that will be deleted.

The Role of Style Files

Collections are configured through style files you choose to include in your style directory. They can be updated continuously while applications are searching over them and document indexing can occur continuously. Collections are automatically serviced for optimization and housekeeping and are automatically repaired when certain error conditions occur (like loss of state information).

- Style files configure the series of indexes storing data about documents and your environment
- The choices made in style files direct how indexing utilities will create indexes and the guidelines for doing work
- Verity products ship with a special style directory which allows you to incorporate many types of documents in your collection, and capture standard field and zone information, without having to do any additional work.

Indexing with vspider

The SEARCH'97 Information Server includes a command-line indexing spider that can create collections from a variety of document types, including:

- ASCII text documents. This includes text documents and documents in Internet e-mail or unused news format.
- HTML or SGML documents
- Adobe Acrobat documents..
- WSIWYG documents. This includes documents supported by Inso's MasterSoft filter and viewer kit, including Microsoft word, WordPerfect and other formats.

The spider can automatically index all of these various document types into a single master collection. Depending on the style files you select, each document may be internally stored in one or more collections. This greatly simplifies collection management and speeds collection information to users.

Following are listed some of the more common commands for building a collection or maintaining a collection using vspider. This is a very helpful utility for routine collection building, for instance at night, when you can set it to run in cron, without having to issue indexing commands directly. All commands must be entered on one line:

Creating a New Collection

```
vspider -collection mycoll.clm -style verity/s97is/locale/english/styles
-start http://www.yoursite.com
```

Re-indexing a Collection

```
vspider -collection mycoll.clm
```

Using domain and exclude options

```
vspider -collection mycoll.clm  -style verity/s97is/locale/english/styles
-start http://www.yoursite.com  -exclude  *secrets*  -domain verity.com
```

Using the proxy option:

```
    vspider -collection mycoll_2.clm -style verity/s97is/locale/english/styles
            -start  http://www.verity.com -proxy proxysvr: 8010
```

Using an authorization file (for password challenge)

```
    vspider       -collection mycoll_3.clm -style
       verity/s97is/locale/english/styles
                  -start  http://www.verity.secure.com/index.htm -auth authfile.txt
```

VSPIDER Options

```
  -style          Style directory (if metaweb add .clm to collection name

  -start          Starting URL or URL's for the spider to follow
                  -start http://www.verity.com   http://www…

  -collection     Path to where the collection is to be created
```

__Indexing options:__

```
  -allold         forces reparsing of all documents in the collection

  -cgiok          allows indexing of URL's containing the ? symbol to permit
  indexing
                  CGI's.  Default is to exclude these.

  -domain         limits indexing to specified domains

  -exclude        excludes files matching a specified expression (exclude *bat
  *exe)

  -host           limits indexing to specified hosts.

  -include        includes files matching a specified expression (include *rpt)

  -maxdocsize     maximum size of document to index  (default is 1 megabyte - enter
                  values in kilobytes)

  -mimeexclude    specifies mime types to be excluded. -mimeexclude  text/html

  -mimeinclude    default is to include all mime types - specify only one

  -nofollow       disables the following of href commands

  -norobo         specifies that any robots.txt files encountered are ignored.
                  The default is to honor all robots.txt files.

  -noupdate       disables updating of any documents already in the collection

  -pathlen        allows you to specify  how far down a URL or directory path the
                  spider should go.  The web host name is not included
                  (www.verity.com:9000/) but elements following it are.  Each slash
                  counts like this/that/then/what would be -pathen 4.
```

-submitsize	specifies the number of documents to be submitted at one time (default is 50)
-topicset	specifies the topicset to be used for indexing the collection
-unlimited	specifies no limits are to be placed on the spider, if neither –host or -domain is specified. The default is to limit based on the host of first URL listed. Also removes the 1 megabyte maxdocsize default unless specified.
-verbose	displays summary information for each URL or file accessed. Indicates indexed, ignored or failed.
-debug	displays debugging-level messages (more than verbose)
-trace	provides highest level of information about indexing
-auth	specifies the name of the authorization file for sites requiring a password
-casesen	makes processing case-sensitive so the spider differentiates by case
-charmap	specifies character map to use
-common	specifies path to VDK home
-datefmt	specifies import format for date Y-M-D M-D-Y etc.
-help	displays syntax options
-language	specifies local language
-jobs	maximum number of sockets to allocate (default is ten)
-msgdb	specifies the path to the ind.msg message database
-noindex	specifies to submit, but not index (used with mkvdk-persist)
-noproxy	specifies the host listed should bypass the proxy server (wildcards ok *.mysite.com)
-proxy	specifies the host &port for the proxy server
-purge	purges the contents of the collection specified
-repair	repairs a damaged collection
-temp	specifies directory for temporary files (default is /tmp)

CREATING COLLECTIONS WITH MKVDK

The process of building a collection with mkvdk is accomplished in three steps:

- Build the collection style
- Locate the documents to be indexed and optionally put their names into a file list (flist).
- Run the mkvdk utility that creates the collection from the style files, for all the documents in the flist, adding other parameters you wish to include to improve searching or performance.

The easiest way to create a new style is to borrow from an existing style and make any necessary changes or additions. The second step, building an flist, can happen in one of many ways. You could just type the names and paths directly into an ASCII file and use that as your flist, or you can use external utility programs that will help you build the list automatically. The third step requires that you run the mkvdk utility with the command-line arguments to create the kind of collection you desire. Housekeeping and optimization are handled automatically as the collection is built.

RUNNING MKVDK

On Windows NT, you can run mkvdk from a command prompt window. The mkvdk utility for Windows is a pure Windows application and cannot be run from DOS so you will need to define properties for it. You can submit as many documents as you have at once, but there is a limit of approximately 60,000 documents per file. Be aware that the usual hourglass is not displayed.

```
c:\verity\platform_dir\bin\mkvdk -create -collection colls\mk_coll
-style styles\wysiwyg c:\docs\text1.txt  (all on one line)
```

The **mkvdk** utility automatically executes the following steps:

1. The specified collection sub-directory and sub-directories are automatically created.
2. The style files used to build the collection are copied from their source directory to the style directory, below the main collection sub-directory and are parsed and compiled
3. The documents are pre-processed and parsed.
4. The documents are indexed and their word data is stored in the indexes.
5. If a Topic set is given, documents are cross-indexed with the Topic set specified.
6. The indexes are then cross-linked and made accessible to the user, word lists and assist data is generated.

INDEXING WITH MKVDK

The mkvdk utility is an all-purpose collection creation and maintenance tool. When you use this utility, you supply the command-line options to tell mkvdk what you want to do. For example, when you are creating a new collection you will provide different options than when you want to do collection maintenance. On-line help with mkvdk is available by entering: **`mkvdk -help`**. The Collection Building Reference Guide includes a full description of each of the mkvdk options. Here are some you will use often.

MKVDK Options

`-create`	`Creates a collection in the specified -collection directory. Creates the directory structure, determines the index contents and sets up the collection based on the style file contents.`

-style *dir*	Specifies the style directory to be used and is only used with the -create option (without this default style files are used)
-description *desc*	Text description of the collection, for example: “Financial Reports”
-words	Builds a word list for each of the individual partitions in the collection
-collection *path*	Specifies the path to the collection to create or open
-nolock	Turns off file-locking (on by default)
-synch	Performs work immediately (rather than default to process in background)
-about	Shows information about the collection including description and last modified
-datapath *path*	Specifies the datapath to use when docs are not located next to the coll directory
-topicset *path*	Creates a topic index based on the named topic set and stores it in the collection directory
-mode *mode* FastSearch, mode Building	Sets the indexing mode. Valid settings are Generic, NewsfeedIdx, NewsfeedOpt, BulkLoad, ReadOnly or any custom defined in the style.plc. Chapter 5 of the Collection manual covers these modes.
-help	Displays syntax options
-debug	Runs mkvdk in debugging mode
-nooptimize	Prevents optimization for this instance
-nohousekeep	Prevents housekeeping (deleting files that are no longer needed) for this instance
-noindex	Prevents indexing for this instance
-charmap	Names the character set that you want all strings mapped to for your application
-locale *name*	Names the language you will use to perform searches (english, deutsch, francais)
-datefmt *format*	Converts a date field into one of Verity’s date representations. Can be used with -extract to affect parsing for date fields. MDY - US format & the default USA - Same as above DMY - European format EUR - Same as above YMD - ISO international format YDM - Swedish format

-servlev *level*	Specifies a particular service: search - enable search & retrieval insert - enable adding and updating documents optimize - enable opportunistic optimization assist - enable building of word list housekeep- enable housekeeping of unwanted files delete - enable document deletion backup - enable backup (duplicate of the collection directory) purge - enable background purging to delete contents without deleting the collection repair - enable collection repair index - same as setting insert & delete dataprep - same as setting search, index, optimize, assist & housekeep

LAB 1: PRACTICING WITH COLLECTIONS

Creating Collections with vspider

1. Under the IS97 directory you will find a main collection directory (colls) and a docs directory which contains several document subdirectories with various types of documents. In these exercises you will build collections from both the vspider utility and the mkvdk utility. As you build each collection, you can create a script (batch file) to capture the various options. It is a good idea to document the features of each of these commands, because you will be taking all of the files you create with you. Please make sure you are always logging verbose output (examples are shown below) for each variation.

2. Click Start and open an MS-DOS Command Prompt window for some practice building collections with **vspider**. In these exercises you will be building collections from the file system. All of your batch files should
be created under **c:\is97\tools.**

   a. Move to the tools directory: `cd c:\is97\tools`

   b. Your first vspider batch file (`vsbld_1.bat`) has been created for you. Open this file and note the documentation about what is being created. Review the command line (using information contained in this section) to identify the options being used, and their purpose. Note that the output is being redirected to a log file (called `vsbld_1.log`) so you will be able to really review the information provided by the indexer.

```
echo off
rem  This batch file creates a collection called vscoll1 under the
rem  IS97\colls directory.  It uses the default style files from the IS97
rem  styles directory and creates a collection of all docs in docs\doc7

vspider.exe -verbose -collection c:\is97\colls\vscoll1.clm (all on one line)
-style c:\is97\styles -start c:\is97\docs\doc7 > vsbld_1.log
```

   c. Run the batch by entering `vsbld_1` at the c:\is97\tools prompt. Remember the output is being written into a log file, so when the directory prompt returns, take a look at the file by entering: `type vsbld_1.log |more` to allow you to page through the log file.

   There is a lot of information in this file about how the indexing works. Note these key points in your log file:

   What version of vspider is running…

```
vspider - Verity, Inc. Version 3.0 (_nti31, May 19 1997)
```

   What you are licensed for…

```
Info      06/17 23:05:52 (ind005005) Licensed for file walking.
Info      06/17 23:05:52 (ind005006) Licensed for local host spidering.
Info      06/17 23:05:52 (ind005007) Licensed for local domain spidering.
Info      06/17 23:05:52 (ind005008) Licensed for remote spidering.
```

   When the work is beginning…

```
Verbose   06/17 23:05:52 (ind002002) VDK: (0 ms) Elapsed Retrieval.
```

The first document is placed in the queue…

```
Verbose   06/17 23:05:52 (ind010003) Submission enqueued: .\docs\doc7\BANKACT.
```

An image document is skipped over - based on its mime-type …

```
Verbose   06/17 23:05:52 (ind020000) Skipping key ..\docs\doc7\BANKACT1.TIF
because of MIME-Type.
```

The next few documents are placed in the queue…

```
   Verbose   06/17 23:05:52 (ind010003) Submission enqueued:
..\docs\doc7\CLAIMS.TXT.
   Verbose   06/17 23:05:52 (ind010003) Submission enqueued:
..\docs\doc7\DEVKIT.DOC.
```

Work is processed…

```
Verbose   06/17 23:05:52 (ind002109) VDK Collection: Processed.
```

Submitted documents information is written to a bulk-submit file…

```
Progress  06/17 23:05:52 (ind031000) Inserting ..\docs\doc7\BANKACT.
Progress  06/17 23:05:52 (ind031000) Inserting ..\docs\doc7\CLAIMS.TXT.
```

Document indexing information is prepared to go into the collection…

```
Verbose   06/17 23:05:52 (ind002002) VDK: BulkInserting into
c:/is97/colls/vscoll1
          (1) from c:/is97/colls/vscoll1/temp/97000000.bif (0, 0).
```

Appropriate partitions are readied (ddd has field values, did has words)…

```
Verbose   06/17 23:05:53 (ind002002) VDK: Initializing dataset 00000001.ddd,
          index 00000001.did.
```

Demographics are captured…

```
Verbose   06/17 23:05:54 (ind002002) VDK: Totals (43 documents): 375 para
          1173 sent  21312 word  (1408 Kb used).
```

The index is populated and automatically optimized…

```
Verbose   06/17 23:05:55 (ind002002) VDK: Optimizing database layout.
Verbose   06/17 23:05:56 (ind002002) VDK: (3655 ms) Indexed 43 docs into
          c:/is97/colls/vscoll1/parts/00000001.
Verbose   06/17 23:05:56 (ind002002) VDK: Writing partition index data.
Verbose   06/17 23:05:56 (ind002109) VDK Collection: Processed.
Progress  06/17 23:05:56 (ind002115) Optimizing VDK collection
          c:\is97\colls\vscoll1.
Verbose   06/17 23:05:58 (ind002109) VDK Collection: Processed.
```

3. One tool we created is the batch file called "**peek**." This calls the browse utility on the partition you just created. Review the commands and then run it by entering ***peek*** at the command prompt. Instructions will be presented on screen. If you want to use this for other collections, you will need to change the collection name on the second line.

```
echo off
cd c:\is97\colls\vscoll1\html\parts

echo Press Enter 3-4 times (each is a record), then type Q to quit
echo The file will be called browse.out - in the tools directory
echo It will display when you are finished...
browse 00000001.ddd > c:\is97\tools\browse.out
cd c:\is97\tools
type browse.out |more
```

4. Another tool we've created is the batch file called "**run-rc**." This runs a special utility program called **rcvdk,** which provides a command line retrieval client. Review the information that is contained in the collection you just built using this handy tool. Before you use it, you will need to edit the SET COLLNAME line to reference your collection name.

```
echo off
SET COLLNAME=vscoll1.clm
cd ..
cd colls
echo This batch file runs a command-line retrieval client program
echo which will allow you to test your collections. Type help to
echo see all of the options or use this primer to get you started.
echo -----------------------------------------------------------
echo a %COLLNAME      = attach to the collection called %COLLNAME%
echo s agent          = search this collection for the term agent
echo s                = perform a null-retrieval
echo r                = show the results list
echo v 1              = view the first document (use # to select)
echo quit             = exit the rcvdk program
echo -----------------------------------------------------------
echo
echo Now we'll ready your collection and you can have some fun!

c:\search97\_nti31\bin\rcvdk %COLLNAME%
```

The following message will appear:

```
Attaching to collection: c:\is97\colls\mycoll
Successfully attached to 1 collection.
Type 'help' for a list of commands.
      RC>
```

Type help to view the list of commands, and practice with these a bit on your collection.

5. Create a new collection (**vscoll2**) and review its contents. Be sure to follow all the steps, making necessary corrections for your collection name, log files and document directories. Use any of the documents under the c:\is97\docs directory -- other than doc7, and use the same style directory that built your first collection.

   a. `Copy vsbld_1.bat to vsbld_2.bat`
   b. Modify appropriately for the new collection name (vscoll2), log file, docs, etc.
   c. Build your new collection and review the log file
   d. Modify the "peek" batch file to point at your new collection and run it, then review the output

e. Modify the “run-rc” batch file to point at your new collection and test it as you explore other options provided by the utility.

Updating and Deleting Documents in the Collection

6. Create another collection (**vscoll3**) for practice with updating and deleting documents. Using vspider, build this collection as shown below, this time directly from the command line. Note that we are turning on advanced logging (by adding **-trace**) to learn more about the indexing process. Again, redirect the output to a file for your review. Enter the following command line (on one line).

```
vspider.exe –verbose -collection c:\is97\colls\vscoll3.clm -style c:\is97\styles
-start c:\is97\docs\doc5 -trace > vsbld_3.log
```

7. Use rcvdk (run-rc) to view your new collection: rcvdk.exe c:\is97\colls\vscoll3.clm **(enter)**

8. Because information does not remain static, vspider comes with utilities, which allow the updating of information already in the collection. In this exercise, you will delete a document from the file system and update the collection. Then you will change a document, update again, and review the changes that have been made to your collection. Remember that your collection was created using the documents in the doc5 directory (c:\is97\docs\doc5).

9. Move to the doc5 directory under c:\is97\docs and physically delete the file called “Partnrs2.doc” from either the command line or from Windows. Run vspider to update the collection as shown below, noting that there is much less information provided on the command line because you are just updating an existing collection.

```
cd c:\is97\colls
vspider.exe -collection vscoll3.clm
```

10. Using what you already learned about rcvdk, take a look at the collection and note the document has been deleted.

11. Move back to the doc5 directory (c:\is97\docs\doc5). Edit the “msg12.txt” file and change the date of the document to today’s date and modify the from field to include your name. Save your changes and run vspider once more to pick up your changes. One important point to understand about the updating process is that it only applies to those documents that were already in the collection. When the update runs, it looks for things that have changed among the collection’s documents, but if you added new documents to this directory, they would not be picked during the update. If you are adding new documents to the collection, you must use the -start flag on the command line and we will do this a little later in this lab. For now, just process your changes:

```
cd c:\is97\colls
vspider.exe -collection vscoll3.clm
```

12. Using rcvdk, search for your name.

13. Update your existing collection (vscoll3), adding the documents from the doc1 directory. To add new documents to an existing collection, you must use the **-start** flag and point to the directory containing the documents (either the same one you used before, or a new directory altogether. For example, if you created a collection for the doc1 and doc2 document directories, then added new documents to the

doc1 directory after the collection was built, you would have to update the collection with the -start flag to catch those documents in the update process.

```
cd c:\is97\colls
vspider.exe -collection vscoll3.clm -start c:\is97\docs\doc1
```

Creating Collections with mkvdk

14. You can also create collections with the mkvdk utility and may need to use this utility to build more specialized collections. You can choose to take advantage of mkvdk's maintenance features, which allow you, for instance, to delete a document from the collection, without removing it from the file system as you did with vspider. For those of you who are only using Verity's Enterprise server, mkvdk will be your only collection building tool.

15. Before using the mkvdk utility, run the edited `mkopt_1.bat` file (in the tools directory) to create a list of command-line options available for mkvdk. When finished, review the associated log file.

16. Review the documents in c:\is97\docs\doc6. You have two special file types here. The first one is a file list called flist.txt. Note that it contains a list of file pointers to the physical location of documents. The second is called hdr.txt. This is a bulk submit file which already contains field attribute information in a text file.

17. As with the vspider practice, we have created your first batch file for collection building with the mkvdk utility, called `mkbld_1.bat`. Review the commands, noting that this batch file uses a file list. You can easily create a file list by typing `dir *.* /b/s>flist.txt`. This creates the flist with full path to your documents. Execute the batch file, and when it finishes, check the log. While there are many similarities to the log you saw from vspider, there are also several differences.

```
echo off
rem  This batch file creates a collection called mkcoll1 under the IS97\colls
rem  directory.  It uses the Verity default style files in the IS97\styles
rem  directory. It creates a collection of some docs in docs\doc6 (those
rem  that are on the file list) and it indexes the collection against the
rem  news topics in IS97\topics.  Verbose messaging is turned on as well.
rem  Note that the @ sign before the filelist specifies to use it as a list.

c:\search97\_nti31\bin\mkvdk.exe -topicset c:\is97\topics\news -verbose
-create -collection c:\is97\colls\mkcoll1 -style c:\is97\styles\autofilt
@c:\is97\docs\doc6\flist.txt >mkbld_1.log
```

18. Execute this batch file and be sure to use "peek" and "run-rc" to view the results. Keep in mind that you will also have to make changes to these batch files to point at your new collection. Once you have access to your collection through RCVDK, try using the expert feature to see more about your documents. Type help and then type X to enter expert mode. Note that this is a toggle and if you type X again, it will revert to novice mode. View field information by entering: `fields title 30 date 11 vdkvgwkey 25`

19. We have included another batch file called mkbld_3.bat that builds a collection from the bulk submit file (called header) in the docs/doc6 subdirectory. Take a moment to review the structure of this file. This file contains all the attribute information, which is to be populated in the index. Note that we are specifying the location in the VdkVgwKey field and that we are using the tag **`<<EOD>>`** to indicate a new

document. If you wish to extract field values from documents, you will find this is the easiest and most efficient way to capture values for your index.

```
VdkVgwKey:   c:\\is97\\docs\\doc6\\rev10.txt
NAME:        Review: Dennis the Menace
DATE:        10/25/93
SOURCE:      Verity Education
<<EOD>>
VdkVgwKey:   ..\\docs\\doc6\\rev11.txt
NAME:        Review: Abbott and Costello Meet Frankenstein
DATE:        10/29/93
SOURCE:      Verity Education
<<EOD>>
```

20. Execute **`mkbld_3.bat`** file. Again, the log file will open after indexing.

```
echo off
rem  This batch file creates a collection called mkcoll3 under the IS97\colls
rem  directory.  It uses the style files copied from the default style area
rem  and creates the collection using a bulk submit file called hdr.txt
echo ...
echo The log file will open when indexing is finished
echo _______________________________________

mkvdk.exe -verbose -create -collection c:\is97\colls\mkcoll3
-style c:\is97\styles\autofilt -bulk c:\is97\docs\doc6\hdr.txt > mkbld_3.log

type c:\is97\tools\mkbld_3.log |more
```

21. Next, delete a document from the collection you just created. The mkvdk utility handles deletes somewhat differently than vspider. With vspider, you actually delete the document from the filesystem and run an update. With mkvdk, you can simply mark the file as deleted, so that it will not be searchable, leaving it on the file system. Move to the colls directory and use rcvdk to view your collection contents:

```
cd c:\is97\colls

rcvdk mkcoll3
s  (to search all)
r  (to view results)
```

Notice the **`rev10.txt`** document. You must enter the file name exactly as it displays in rcvdk. This means that if you had been deleting another document on the list, you might enter the file specification differently (relative as opposed to absolute). The VdkVgwKey is determined by how you build the collection or by the entries in a bulk submit or file list. In our example, we have entered some documents using relative path and others using absolute. Delete the document from the collection with the following command:

```
mkvdk -collection c:\is97\colls\mkcoll3 -delete c:\is97\docs\doc6\rev10.txt
```

After it's been deleted, use rcvdk again to check the results. Keep in mind that deleting documents requires servicing the collection and it may take a couple of moments to see the update. The first time you had 5 documents in the collection, when you attach and search again, you should have only 4. When you view the results, you will see the document has been deleted from the collection, but when you go to the directory, you will see the document still exists.

22. Finally, update a document from this same collection. The mkvdk utility also handles updates differently than vspider. With vspider, when there are any changes: additions, deletions or modifications, these are “sensed” when the collection is serviced. With mkvdk, it is a process of notifying the collection to do an update to a particular document or list of documents.

```
cd c:\is97\docs\doc6
```

Edit the `rev11.txt` document. Add something to the name of the movie (& your name). Save and exit the document. Note the difference in how the file specification is written.

```
cd c:\is97\colls
mkvdk -collection c:\is97\colls\mkcoll3 -update ..\docs\doc6\rev11.txt
```

When mkvdk is finished, use rcvdk to attach to your collection and search for whatever you added to the document.

In these exercises you have had the opportunity to practice with the collection building and maintenance tools provided by the two primary Verity indexers. The choices you make about which options and which style files to use are based on the features you want to support in your collections.

If you have been working with Verity products for some time, you may have used other indexers and you may already have collections built that you would like to add to your SEARCH’97 Enterprise Server. You do not need to rebuild your existing collections, as long as they provide all the functionality and features, you desire. If you want to take advantage of new features or the simplification provided by the universal filtering capabilities, you will need to rebuild your collections.

EXPLORING STYLE FILES

This section provides details on style file creation, including:

- The Role of Style Files
- Defining Fields
- Creating the Virtual Document
- Filtering and Formatting by Zones
- Advanced Style File Features

The Role of Style Files

The primary role of style files is to provide you with an opportunity to customize your view of document information. Organizational needs differ regarding the fields that are important and how the data for those fields should be stored. Each style file allows the configuration of some important component in your collection and you may choose a variety of different styles for groups of documents at your site. Each collection can be defined with a unique style, but only one set of style files can be used.

Style files needed by the collection are described below. Some style files are mandatory for the collection to be built, while others are optional and used only when needed for specific functionality.

The collection requires all style files be named the same. The root name is style, followed by the appropriate file extension (.ddd, .dft). When you create a collection you specify which style files will be used. As the collection is created this style directory is copied locally to the collection directory.

This process serves two purposes. First, by moving a copy of the style files used to build the collection into a directory contained in the collection itself, you need not be concerned about losing or accidentally deleting important style files. Second, including a copy of the style files helps to make collections portable. Collections are designed to allow you to move them to another disk, another server, or even another directory, if needed.

EXPLORING STYLE GUIDELINES

You may not need to alter any style files. Verity tries to anticipate the kinds of information you will wish to extract from documents and do the prep work for you. However, it is possible that your documents will have special information, which could not be anticipated. When this is the case, you will need to edit style files to add the information you want to extract.

Just to give you an idea about how the style files are defined, here are a few that are used in the main style directory provided by Verity.

THE STYLE.DDD FILE CONFIGURES THE COLLECTION

```
# $Id: style.ddd, v 1.1.1.5 1997/02/23 04:19:02 wade Exp $
# Copyright (C) 1987-1997 Verity, Inc.
# Document Dataset Descriptor
$control: 1
$include style.prm
$subst: 1
descriptor:
  /collection = yes
```

```
{
  data-table: _df
    /num-records = 1
    /max-records = 1
  {
    # Header information for partition management
    worm:        _DBVERSION       text
    fixwidth:   _DDDSTAMP        4 date
    varwidth:   _DOCIDX          _dv
    fixwidth:   _PARTDESC        32 text
    constant:   _FtrCfg          text "${DOC-FEATURES:}"
    constant:   _SumCfg          text "${DOC-SUMMARIES:}"

    fixwidth:   _SPARE1          16 text
    fixwidth:   _SPARE2          4 signed-integer
  }
  # Required internal fields per document
  data-table: _df
    /offset=64
  {
    autoval:     _STYLE           sirepath
    fixwidth:    _DOCID           4 unsigned-integer
    fixwidth:    _SECURITY        4 unsigned-integer
      /minmax = yes
    fixwidth:    _INDEX_DATE      4 date
      /minmax = yes
  }
$ifdef DOC-FEATURES
  # Optional feature vector per document
  data-table: _dg
  {
    varwidth:   VDKFEATURES _dh
      /_implied_size
  }
$endif
$ifdef DOC-SUMMARIES
  # Optional generated summary per document
  data-table: _di
  {
    varwidth:   VDKSUMMARY      _dj
      /_implied_size
  }
$endif
$include style.sfl
$include style.ufl
}
$$
```

THE STYLE.DFT FILE DEFINES THE VIRTUAL DOCUMENT

HTML Version

```
# $Id: style.dft, v 1.1.1.5 1997/04/17 00:04:43 ameyer Exp $
# Copyright (C) 1987-1995 Verity, Inc.
# Document Format
#
$control:1
dft:
{
  field: DOC
    # The following line invokes the HTML zone filter
    /filter="zone -html"
}
$$
```

Auto-Filt Version

```
# $Id: style.dft, v 1.1.1.1 1996/09/12 22:46:56 ameyer Exp $
# Copyright (C) 1987-1995 Verity, Inc.
# Document Format
#
$control:1
dft:
{
  field: DOC
    # The following line invokes MasterSoft WYSIWYG filtering
    /filter = "auto"
}
$$
```

PDF Version

```
# $Id: style.dft,v 1.1.1.3 1997/02/28 19:14:57 wade Exp $
# Copyright (C) 1987-1996 Verity, Inc.
# Document Format
#

$control:1
dft:
{
  field: DOC
    /filter="flt_pdf -charmapto 850"
}
$$
```

THE STYLE.PRM FILE GOVERNS WORD INDEXING FEATURES

```
# $Id: style.prm,v 1.1 1997/02/23 04:26:15 wade Exp $
# Copyright (C) 1987-1995 Verity, Inc.
#
# style.prm - collection schema parameters
#
# This file is used to enable/disable index schema features through
# macro definitions similar to those allowed by the C preprocesser.
# This file is included in other style files using $include so
# that the selected features are propagated to the schemas of all
# tables in the index.  Refer to the "Using the style.prm File"
# chapter in the Collection Building Guide for more information.

# ------------------------------------------------------------------
# The IDX-CONFIG parameter defines the storage format used to
# encode the word positions in the index.  WCT (Word Count) format
# is a compact format, storing the ordinal counting position of the
# word from the beginning of the document.  PSW (Paragraph, Sentence,
# Word) format takes approximately 15-20% more disk space, but
# stores semantically accurate paragraph and sentence boundaries.
# Optionally, Many may be specified with either WCT or PSW to
# improve the accuracy of the <MANY> operator at the expense of
# diskspace and search performance.

# This example enables Word Count word position format (the default).
$define                 IDX-CONFIG       "WCT"

# This example turns on Paragraph/Sentence/Word word position format.
# It also enables the <MANY> operator accuracy improvement.
#$define IDX-CONFIG    "PSW Many"
```

```
# ------------------------------------------------------------------
# The IDXOPTS parameters define which index options are applied to
# the various index token tables.  The following index options are
# supported for each: Stemdex enables an index by the stem of each
# word. Casedex stores all case variants of a word separately, so
# one can search for case sensitive terms such as "Jobs", "Apple",
# and "NeXT" more easily. Soundex stores phonetic representations
# of the word, using AT&T's standard soundex algorithm.  The
# application may also store 1-4 bytes of application-specific
# data with each word instance, in the form of Location data and/or
# Qualify Instance data.  These options are specified separately
# for each token table: word, zone, and zone attribute.
$define                WORD-IDXOPTS     "Stemdex Casedex"
$define                ZONE-IDXOPTS     ""
$define                ATTR-IDXOPTS     "Casedex"

# ------------------------------------------------------------------
# Clustering is enabled by uncommenting the DOC-FEATURES line.
# This stores a feature vector for each document in the
# Documents table.  These features are used for Clustering
# results and fast Query-by-Example.  See the discussions on
# Clustering in the Collection Building Guide for more information.
$define DOC-FEATURES    "TF"

# ------------------------------------------------------------------
# Document Summarization is enabled by uncommenting one of
# the DOC-SUMMARIES lines below.  The summarization data is
# stored in the documents table so that it might easily be
# shown when displaying the results of a search.
# See the discussions on Document Summarization in the
# Collection Building Guide for more information.
# The example below stores the best three sentences of
# the document, but not more than 500 bytes.
$define DOC-SUMMARIES   "XS MaxSents 3 MaxBytes 500"
# The example below stores the first four sentences of
# the document, but not more than 500 bytes.
# $define DOC-SUMMARIES   "LS MaxSents 4 MaxBytes 500"
# The example below stores the first 150 bytes of
# the document, with whitespace compressed.
# $define DOC-SUMMARIES   "LB MaxBytes 150"
```

THE STYLE.WLD GOVERNS HOW WORDS ARE STORED

```
# $Id: style.wld,v 1.1.1.3 1997/04/02 23:26:47 wade Exp $
# Copyright (C) 1987-1997 Verity, Inc.  style.wld - Word List Descriptor
#
$control: 1
$include style.prm
$subst: 1
descriptor:
{
  data-table:           _ph
    /num-records=1
    /max-records=1
  {
    # Header information for partition management
    worm:                   _DBVERSION     text
    fixwidth:   _DDDSTAMP              4  date
    fixwidth:   _DIDSTAMP              4  date
    constant:   Types           text "Word Zone Attr"
    constant:   Config          text "$IDX-CONFIG"
    constant:   Word            text "$WORD-IDXOPTS"
    constant:   Zone            text "$ZONE-IDXOPTS"
    constant:   Attr            text "$ATTR-IDXOPTS"
    constant:   TSPARE1         text ""
    constant:   TSPARE2         text ""
    constant:   TSPARE3         text ""
  }
# This is the table of parts covered by spanning wordlist
  data-table:          _pp
  {
    fixwidth:          PARTNUM     4 unsigned-integer
  }
  # The Actual full text index data
  data-table:   _pf
  {
    varwidth:   FWTEXT          _pw
      /_implied_size
    varwidth:   FWDATA          _pv
      /_implied_size
    fixwidth:   FWENCODE        1 unsigned-integer
    fixwidth:   FWFREQ          2 unsigned-integer
  }
# The Btree for fast word lookup
  data-table:   _pb
  {
    fixwidth:   FWBTREE         3 text
  }
# The accelerator for lookup by Stem
  data-table:   _ps
  {
    fixwidth:   STEMDEX         4 unsigned-integer
  }
# The accelerator for lookup by Soundex
  data-table:   _px
  {
    fixwidth:   SOUNDEX         4 unsigned-integer
  }
}
$$
```

THE STYLE.SFL FILE GOVERNS STANDARD FIELD DEFINITION

```
# $Id: style.sfl,v 1.4 1997/04/19 01:47:13 bjohnson Exp $
# Copyright (C) 1997 Verity, Inc.
#
# style.sfl - Verity-Defined Standard Fields
#
# These fields are included in the internal documents table.
# They are filled in by various filters and gateways that Verity ships,
# and are the "standard fields" that Verity suggests should exist in all
# Verity collections. They are not required in your collection. Instead,
# they are merely highly recommended to promote the ability to use your
# collection with other products that use Verity's search technology.
# You can comment out the fields below to save space, or uncomment others
# to gain functionality.
#
# Note that "flt_inso" refers to the "auto" filter that can use either
# the Mastersoft or Inso OutsideIn filters.
#
data-table:   _sf
{
```

```
# Title is filled in by: zone -html, flt_pdf, zone -email, zone -news
 varwidth: Title     _sv
                      /alias = FTS_Title
                      /alias = Subject

# Author is filled in by: flt_pdf, zone -email, zone -news, flt_inso
varwidth: Author     _sv
                      /alias = From
                      /alias = FTS_Author
                      /alias = Source

# Keywords is filled in by: flt_pdf, zone -news, flt_inso
varwidth: Keywords   _sv
                      /alias = FTS_Keywords
                      /alias = Keyword
                      /alias = Comments
                      /alias = Reference

# Snippet is filled in by: vgw_url, flt_inso
varwidth: Snippet    _sv
                      /alias = Abstract

# MIME-Type is filled in by: universal
varwidth: MIME-Type  _sv

# Charset is filled in by: flt_cmap
varwidth: Charset    _sv

# To is filled in by: zone -email, flt_inso
varwidth: To                   _sv
                     /alias = Destination

# Date is filled in by: zone -email, zone -news, flt_inso, flt_pdf
#
# This field is the "last modified" date, not the creation date
fixwidth: Date        4 date
                     /alias = Modified
                     /alias = FTS_ModificationDate
                     /alias = Recorded_Date
                     /alias = Version_Date

# NewsGroups is filled in by: zone -news
varwidth: NewsGroups _sv

# PageMap is filled in by: flt_pdf
# This field is required to do highlighting in pdf documents. Do not
# comment this if you want pdf highlighting!
varwidth: PageMap    _sv
  /_hexdata = yes

# The following are fields that could be filled in by Verity shipped
# code, but they are currently not populated by many documents. To
# save space, they are commented out here. You may comment them back
# in before indexing your documents if you need them in your collection.
#
# The following fields are filled in by "zone -news"
#varwidth:   References        _sv

# The following fields are filled in by flt_inso
#varwidth: Last_Author         _sv
#varwidth: Account             _sv
#varwidth: Address             _sv
#varwidth: Attachments         _sv
```

```
#varwidth: Authorization       _sv
#varwidth: Backup_Date         _sv
#varwidth: Bill_To             _sv
#varwidth: BCC                 _sv
#varwidth: CC                  _sv
#varwidth: Category            _sv
#varwidth: Checked_By          _sv
#varwidth: Client              _sv
#varwidth: Completed_Date      _sv
#varwidth: Character_Count     _sv
#varwidth: Page_Count          _sv
#varwidth: Word_Count          _sv
#varwidth: Created             _sv
#varwidth: Department          _sv
#varwidth: Destination         _sv
#varwidth: Disposition         _sv
#varwidth: Division            _sv
#varwidth: Minutes_Edited      _sv
#varwidth: Editor              _sv
#varwidth: Forward_To          _sv
#varwidth: Group               _sv
#varwidth: Language            _sv
#varwidth: Last_Print_Date     _sv
#varwidth: Mail_Stop           _sv
#varwidth: Matter              _sv
#varwidth: Office              _sv
#varwidth: Operator            _sv
#varwidth: Owner               _sv
#varwidth: Project             _sv
#varwidth: Publisher           _sv
#varwidth: Purpose             _sv
#varwidth: Recorded_By         _sv
#varwidth: Recorded_Date       _sv
#varwidth: Modified            _sv
#varwidth: Revision_Notes      _sv
#varwidth: Revision_Number     _sv
#varwidth: Secondary_Author    _sv
#varwidth: Section             _sv
#varwidth: Security            _sv
#varwidth: Source              _sv
#varwidth: Status              _sv
#varwidth: Document_Type       _sv
#varwidth: Typist              _sv
#varwidth: Version_Date        _sv
#varwidth: Version_Notes       _sv
#varwidth: Version_Number      _sv

# The following fields are filled in by flt_pdf
#varwidth:   FileName          _sv
#fixwidth:   NumPages          4      unsigned-integer
#fixwidth:   PermanentID       32     text
#fixwidth:   InstanceID        32     text
#varwidth:   DirID             _sv
#fixwidth:   WXEVersion        1      unsigned-integer
#varwidth:   FTS_Creator       _sv
#varwidth:   FTS_Subject       _sv
#varwidth:   FTS_Producer      _sv
#fixwidth:   FTS_CreationDate  4   date

# The following fields are filled in by vgw_url
#varwidth: Ext                 _sv
#varwidth: URL                 _sv
#varwidth: _Created            _sv
```

```
  #varwidth: _Modified           _sv
  #fixwidth: Created             4      date
  #fixwidth: Modified            4      date
  #fixwidth: Size                4      unsigned-integer
}
```

THE STYLE.UFL FILE GOVERNS USER DEFINED FIELDS

```
# $Id: style.ufl,v 1.1 1997/02/23 04:26:39 wade Exp $
# Copyright (C) 1987-1997 Verity, Inc.
# style.ufl - Application-specific User Fields
#
  data-table:          dkf
  {
    # Must have a VdkVgwKey - this is the primary key to the document
    # The field type can be changed to fixwidth integer or text as needed
    varwidth:          VdkVgwKey   dkv
      /indexed = yes
      /minmax  = yes
  }
  data-table:          ddu
  {
    varwidth: Workgroup          ddh
    varwidth: Product            ddh
  }
```

FIELD DEFINITIONS STYLE.DDD

The style.ddd file defines the fields that you want to create for the collection. All field names and values are stored in the Document Dataset. You can create additional fields with a constant value for your virtual documents. These constant fields are those that appear in each virtual document and have the same field value for each document. The constant field values are added to the Document Dataset when you build the collection, based on instructions you place in the style.ddd file.

The collection captures information about the documents that it contains. For each document, the document database stores key information such as its title, author, and date. Additionally, the database stores information about the location and size of the document, since collections do not physically store documents within the database architecture. You decide which fields you want to create and then take advantage of the fields as desired. Defining helpful fields such as title and author makes your information application much easier to use. It is important to understand that all documents do not have to contain all of the fields you define. Documents without some fields will simply not be scanned when performing field searches.

There are three basic types of fields that can be defined in a style.ddd file. These statements are stored in special data-tables to improve performance. Data-table names should be three characters in length. Field types that you may include are:

Static Value Fields (constant, autoval, worm):

Constant value fields assume the same for the entire collection. They can be up to 128 characters in length. If the constant value includes white space, you must enclose it in quotes. The format is to list the field name, the data-type, and the value:

```
        constant: DBDOC text "Lessons Learned"
```

This statement says to create a new field called DBDOC for the collection and to populate the field with the text string "Lessons Learned. This data is only stored once.

Variable Fields (fixwidth, varwidth):

Variable fields are populated with variable values parsed from the original document text according to Verity's formatting and filtering rules for automatic extraction of zones and fields or for rules in a style.tde file. Variable fields can have modifiers that create a separate index for fields within the Document Dataset or limit field searches to those that are case-sensitive.

```
fixwidth: DATE 4 date

varwidth: TITLE dd1
    /indexed = yes or no
    /case-sensitive = yes or no
```

Index options increase performance during the field level searches. Create these indexes only for the fields that will be regularly searched.

If field-level searches are to be conducted over a field containing a date, the date field must be defined as a fixed-width field and assigned a data type of date. You will always use a length of 4 for date fields because they are translated by an internal program into a special Verity date format. This special date format allows documents to represent dates in a variety of ways and still be searched or sorted appropriately. It also allows users to enter dates in a variety of ways and find their information regardless of the format that exists in the document.

Document Dispatch Field:

The dispatch field statement identifies a special field that is used when a user wishes to display a document. In most cases you will define the field DOC to display the entire document on screen. If you wish to display only part of the document, you may indicate the start-of-document address pointer (to any point where you wish to begin displaying text in the document window) in your style.dft file.

```
dispatch: DOC (or field name)
```

CONFIGURING THE VIRTUAL DOCUMENT: STYLE.DFT

The style.dft file contains the rules for displaying for the virtual document. The default virtual document consists of one field called doc (which is your entire document). This document appears beginning in row 1 and column 1 of the document window.

The virtual document identifies document filters and formatting used for indexing and displaying documents. These default settings are enabled:

- The Universal Filter which automatically detects all supported document types using the helper filters
- The WYSIWYG Filter which supports searching over binary documents and which are incorporated in the KeyView Filter Kit
- The Zone Filter which supports searching over regions of text defined as zones
- The PDF Filter which supports searching of Adobe Acrobat PDF documents
- A generic document layout definition which indicates the body text to be indexed and viewed

A style.dft file allows you to change the appearance and contents of the virtual document. You can change margins and spacing of fields, select field values, add standard or repetitive information, or incorporate text contained in a separate file.

The dft Statement

The dft statement identifies the file as a style.dft file. This statement specifies how data is to be dispatched to the screen when a user chooses to view the document.

The dft statement may have three modifiers:

/fill	create a newline if a newline character is encountered in a field value or constant(default = no preserves new lines in the document)
/right-margin	defines right-margin for document display (default = 70)
/tabsize	defines number of indent characters to be used when a tab is encountered (default = 8**)**

Miscellaneous style.dft Statements

Keyword statements under the dft statement include:

field	name of field defined in style.ddd that you wish to display in each document window
constant	string of text that you want displayed in each document window
file	full or relative path to a document that you want displayed in each document window
system	specifies a system call that is to be done before Topic displays information in the document window. The result of the system call can be a part of the display.

Each of these statements may have the following modifiers. Note that some of these modifiers are the same as those used above in the dft statement. Modifiers for the dft statement are global variables for all keyword elements.

/fill	create a newline if a newline character is encountered in a field value or constant (default = no preserves new lines in the document)
/right-margin	defines right-margin for document display (default = 70)
/tabsize	defines number of indent characters to be used when a tab is encountered (default = 8)
/filter	specifies the I/O filter_name that will convert the native format of documents
/charmap	defines the character map to be used to print characters to the screen when a foreign language is being used. There are two choices: 850 (IBM code page 850) or 8859 (ISO-8859)
/row	specifies the row number in which the field value or string will be displayed
/col	specifies the column number in which the first character of the field value or string will be displayed
/delta-row	specifies the row number relative to the text above it where you want the field value or string to be displayed

/delta-col	specifies the column number relative to the right-most character in a row where you want the first character of a field value or string to be displayed

A Basic style.dft File

```
$control: 1
dft:
    /right-margin = 60
{
    constant: "Gardening Journal"
        /row = 2
    field: TITLE
        /row = 4
        /col = 10
    field: DOC
        /delta-row = 2
}
$$
```

The document window, using this style.dft would look like this:

```
Gardening Journal

          Planting Onions in Winter

I have had a lot of experience planting sweet onions.
Though onions are thought to be a seasonal crop, I've been
experimenting with greenhouse onions. To my great surprise,
the onions I planted this winter were the best ever.
```

DEFINING ZONES

Zone searching allows you to define regions in a document where restricted searching can be applied. For example, your documents may contain an abstract field or a subject keywords field. Finding your search term in one of these regions might prove your subject better than finding it anywhere in the general text of the document. Zone searching also produces faster results than traditional field searching. One caution is that zones cannot do range searching, so dates defined as zones would not typically make sense. Another caution is that the traditional field operators cannot be used either. Queries against zones use the operator <IN>. For example, **`John <in> publisher`**. You can define a region of text as both a field and a zone. This allows differing types of searching to be effective based on the user's requirements. It can, however present some new training issues to make sure your users understand how they should query.

The key difference in zone searching is how and when parsed information is processed. In the traditional model, indexing handles up front parsing and the creation of a bulk-insert (header) file for processing. Even if you still follow this model, the values may be changed when you use zones. The reason for this is that zones are populated at indexing time based on definitions in your dft.

Zones also require well formatted documents and were designed primarily for HTML and SGML documents, and with a bit of creativity, even ASCII documents, can use features in the style.dft to create custom zones.

INDEXING WITH ZONES

There are three built-in zone modes and you must specify one with the zone argument

- -html filters hypertext markup language
- -email filters Internet email conforming to RFC822 standard
- -news filters Internet Usenet news conforming to RFC822

By default the SEARCH'97 engine invokes the universal filter with the zone filter as a helper or you can specify the zone filter in a style.dft file. If you place a filter spec in the style.dft, the collection must be limited to those type of documents.

When the zone filter encounters a start tag, it opens a new zone and closes it when it encounters an end tag. The indexer then makes a zone out of all the text between the two tags. The tags themselves are ignored and tag names are case-insensitive.

The zone filter recognizes most standard tags through HTML 3.0 It automatically extracts these tags as fields and ignores the rest

```
<a>            <code>    <head>
<abbrev>       <dfn>     <html>
<acronym>      <fn>      <lang>
<address>      <form>    <link>
<au>           <h1>      <note>
<banner>       <h2>      <person>
<base>         <h3>      <q>
<blockquote>   <h4>      <samp>
<body>         <h5>      <textarea>
<cite>         <h6>
```

LAB 2: EXPLORING STYLE FILES

1. Under the IS97 directory you will find a main styles directory which contains a number of style subdirectories. The most important style directory to review is the one called style. This is the one that takes the most advantage of SEARCH'97 features. Move to this directory and review the following key features in each of the
style files.
    a. Edit the style.ddd
    b. Note the comments at the top of the file about not adding user fields to this file. Beginning with SEARCH'97, the style.ddd file (which used to hold all information about fields in the collection) has a new structure. The idea is to use the .ddd as a gatherer of other style information - which includes
    the style.sfl - for all the fields that Verity has learned are important to customers, and your style.ufl –where you add any other fields you want to include.
    c. A few lines down you will find the style.prm. This style file governs how you prefer to store word index information. Notice that it is called with the $include statement.
    d. As you read through this file, you will find many fields being created by Verity, which are required to make the collections work properly. You should also notice the structure of this file. Organized in data-tables, you will find almost every type of field in this file. These are some of the fields you see when you "browse" a collection.
    e. At the end of the file, you will see that there is another $include to the style.sfl (Verity's standard field list) and to the style.ufl. Do you remember what the style.ufl is for?
    f. Close the .ddd and open the style.sfl. This style contains all of the standard fields that will be captured by Verity's filters and gateways. This is a well documented file, so take a few minutes to its contents and structure. Note too that there are many additional fields you could use, by simply uncommenting them.
    g. Close the .sfl and open the style.ufl. What is contained here?
    h. The top of the style.ufl provides some quick training on how to add your own user fields.
    i. Close the .ufl and open the style.dft. As you can see, this .dft is calling the universal filter. What does the .dft configure?
    j. Close the style.dft and open the style.uni. What is contained in this style file? Review the various components and comments and note the different ways applications are covered based on mime-type.

2. Open a second Command Prompt Window so you can view other style files, side by side. Return to the main style directory and wind your way through the various directories. Notice that they have different contents, different numbers of style files and many examples. Watch for some that have unique files like style.lex, style.zon, and style.tde. After this lab, we will explore some of these unique style files and learn about customization alternatives for collections.

CONFIGURING THE WORD INDEX

Three additional style files allow you to refine your word searching. The use of any one of these style files affects the contents of the collection's document index. If you use a style.lex file, your document index will grow to accommodate all words that include any of the special characters you have specified. If you use a style.stp to stop words from appearing on your word index, the document index will be smaller. Rarely does anyone use the style.go file, which limits your index to only those words contained in the file.

THE STYLE.LEX FILE

When you wish to include non-alphanumeric characters such as ampersands or slashes as part of your words, you will need to specify these characters in a lex file. When you include a lex file in the style sub-directory, the indexer will use your lex definition in place of its own internal lexical analyzer. It will take a little longer for the build when using your own lex file because it is not integrated into the program. As document indexing runs, you will see messages telling you that the indexer is reading the document lexer, compiling, and writing the new document lexer for the collection you are creating. If you use this same set of style files to create another collection, the compiled style.lxt version will be used.

Since the indexer will use your lex file in place of its own, you must define all the key components in any lex file you create.

SAMPLE STYLE.LEX FILE

```
#
# style.lex
#
$control: 1
lex:
{
    define: WHT          "[ \t]"
    define: NL           "{WHT}*\n"
    token:  WORD         "[A-Za-z0-9]+"      # word
    token:  WORD         "[0-9]+\\.[0-9]+"    # word
    token:  EOS          "[.?!]"             # end of sentence
    token:  NEWLINE      "{NL}"              # single end-of-line
    token:  PARA         "^([ \t]*\n)+"      # end of paragraph
    token:  WHITE        "{WHT}"             # whitespace
    token:  PUNCT        "."                 # all other text
}
$$
```

THE STYLE.STP FILE

You can exclude frequently used words from your word index by adding a style.stp file to the style directory before building your collection. A style.stp file is a regular ASCII file where you list the words you do not want to include in your document index. This file is case-sensitive, so if you want all variations of words, you will need to list them individually or define them by regular expression. Be sure to inform your users about words you have excluded. Remember that Verity products search for matches against the index. If a user queries on "the white house" and the word "the" has been excluded, no results will be returned. This can be very confusing for end users and for this reason, we generally recommend against the stopping of words.

Viewing Words in the Index

You can use the didump utility to see words stopped in a collection. The didump utility includes information about the occurrences of words in a single partition. Output can be viewed on the screen or redirected to a file.

If words have been auto-stopped, you may find a value of 0 in the first four columns of the index. If you are concerned that a particular word might occur frequently and wish to check the status, you can use the didump utility and specify the word.

For instance, from the coll_a collection, you could use the didump to see the occurrences of the word "the" in the word indexes, which are stored under the parts directory.

```
didump -pattern the 00000000.did

      Word  Size  Doc   Word
      THE   8     1     1
      The   98    5     27
      the   605   6     195
```

THE STYLE.GO FILE

The style.go file is also an ASCII file that lists words, one word per line, which will be the only words to be indexed. As this runs counter to the whole idea of text retrieval, it is rarely used. Format of the style.go file is exactly the same as the style.stp file.

WHAT IS A "POLICY FILE"?

To understand what a policy file is, and why it would be needed, we must first take a few steps backward and look at the bigger picture. Verity applications are products that are built on Verity's SEARCH'97 Developers Kit Engine. This engine is advanced and flexible.. One of the benefits of the engine is that it handles most of the work handed to it in real-time, which means that as soon as the engine is done processing its work, the users are able to see the results of that work.

In order to build a system that allows for users to be reading while the engine is writing, a controlling component had to be put in place to deal with "traffic" and make sure that all work requests (reading and writing) were taken care of. This "traffic manager" component is referred to as the service loop. The service loop has many different tasks it must execute. These tasks and their associated rules, are controlled through the policy file. Default values for each of these parts are built into Verity products, but through the use of a policy file, you can override these default values. Section 5 of the Collection Building Guide addresses the features and settings of the policy file.

ADDITIONAL MATERIALS SECTION

- **CLIENT SERVER APPLICATION: ENTERPRISE SERVER/TOPIC AGENTS**
- **REGULAR EXPRESSIONS**
- **STYLE.TDE FILE**

THE ENTERPRISE SERVER

The SEARCH'97 Enterprise Server offers access to collections, knowledgebases, and search technology in a client/server application. There are two types of connections that can be made from client products, Local and Remote. A local connection means reading and searching information in collections and knowledge bases that are visible to the machine running the client via some kind of drive letter or path. This does not (as it's name implies) mean that the information must actually live locally, just that it can be seen via a path. For instance: a collection may be placed in **`C:\DEMOS`**, which is on the machine's local hard drive, or it could be placed in **`R:\COMMON\DATA`**, which is a directory on a LAN server.

Remote connections, on the other hand, do not require that the machine be able to see the collection via any kind of path. With a remote connection, client products use IP (Internet Protocol) to connect with an external computer where an Enterprise Server has been installed and configured. Once this work has been done, users can submit their search requests to the server and receive their results from the server.

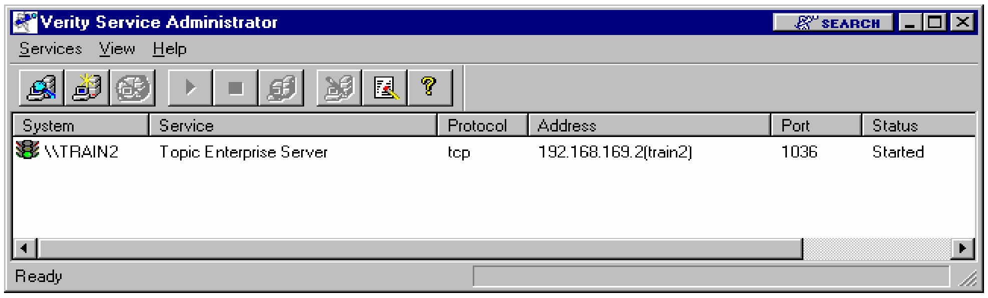
To make a remote connection, the server must be set up and listening to a port for search requests. The server is basically a daemon that is run from a UNIX or Windows NT command line prompt. A configuration file for the server is created and modified to suit the needs of the clients it will serve. This config file includes information about which collections and knowledge bases to serve, what port to listen to requests for, how many users are allowed to connect, etc..

Multiple servers with different sets of collections and knowledgebases can be set up to watch different "ports," serving selected information in different configurations. For example, server daemon "A" could be set to serve a NewsWatch on port 8005 while server daemon "B" is giving out sports scores on port 8006. These are two different instances of the same program, run with two different configuration files. This kind of set up provides great flexibility in specifying what, where, and how your information will be served.

Before you start your server, you should have created collections and topicsets that will be provided through the server port. Once your Enterprise Server is installed, you will find sample configuration files for every platform in the admin subdirectory. The first thing you need to do is to edit the appropriate configuration file for your platform. At a minimum you will need to set your server spec (tcp:5000) and make sure your startup collection and startup knowledgebase are properly pathed.

STARTING THE SERVER

You can start the server from a command line or from an icon. The SEARCH'97 Enterprise Server for NT includes a graphical Verity Service Administrator that allows you to start and stop your server, make changes to the configuration file, and monitor activities.



System	Service	Protocol	Address	Port	Status
\\TRAIN2	Topic Enterprise Server	tcp	192.168.169.2(train2)	1036	Started

STARTING THE CLIENT

You can connect to the Enterprise Server via Topic Agents. In either case you will make a remote connection, supplying the port information for your server. Topic Agents allow you to create an agent that captures user preferences for the remote connection.

THE IMPORTANT ROLE OF MAP FILES

Two important components in the configuration of clients are the specifications for which collections are to be used and which topic sets (knowledgebases) will be used. Both of these are configured in a special **map file** which describes how users will see the appropriate components and which will be included. When the user logs into the client through the server, these map files direct the user's access to information.

THE COLLECTION MAP FILE

A collection map file is used to describe the collections you will make available to your users. It includes the order and relationship in which collections appear, as well as the path to the physical collection. You can use virtual names for the collections, which are then mapped, to their physical locations. A collection list map can "call" other collection list map files – as long as the total number of collections does not exceed 128.

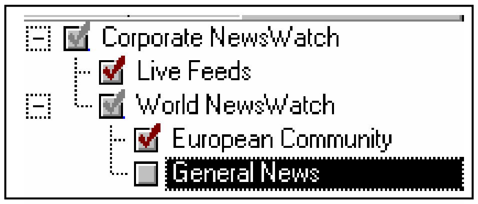
Sample Collection Map (.clm) File

```
$control: 1
virtual-collection: "Corporate NewsWatch"
{
      node:         "Live Feeds"
                    /collection-path = c:\is97\colls\newsfeeds

      node:         "World NewsWatch"
      {
         node:      "European Community"
                    /collection-path = c:\is97\colls/ec

         node:      "General News"
                    /collection-path = c:/verity/colls/misc
      }
}
```

Using the collection map shown above, a Topic Agents user would see the following structure when they click on the Collections tab and expand the collections:



Users can then select at any node level to enable documents for searching. The DBA should, therefore, focus on which collections are logically going to be desirable to search as a group and which are more likely to be standalone. By defining them hierarchically in this file, users can select the top level node with one click, turning on all collections, which reside below.

Ideally, your collection map files will be in a central location so that you have only one file to manage when adding new collections for your users.

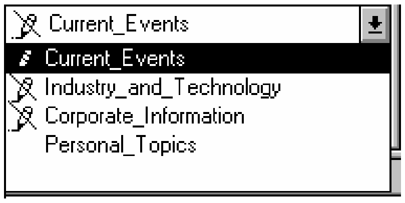
A knowledgebase map file is used to describe the topic sets you will make available to your users. It includes the order and name you wish to display, as well as the path to the physical topic set. You can use virtual names for the topic sets, which are then mapped, to their physical locations.

Sample Knowledgebase Map (.kbm) File

```
$control: 1
kbases:     "Corporate NewsWatch"
{
      kb:   "Personal Topics"
            /kb-path = /users/joe/personal.skb
      kb:   "Corporate_Information"
            /kb-path = toplib/corp
      kb:   "Industry_and_Technology"
            /kb-path = toplib/int
      kb:   "Current_Events"
            /kb-path = toplib/news
}
```

note:
no spaces in names

Using the knowledgebase map shown above, a Topic Agents user would see the following structure when they click on the Topics tab and select their knowledgebase:


Current_Events
Current_Events
Industry_and_Technology
Corporate_Information
Personal_Topics

The icon above indicates that most of the topic sets cannot be written to. The Personal_Topics knowledgebase is the only one that can be updated. Note that the order shown is the exact opposite of the way the file is defined. Each knowledgebase can hold approximately 20,000 topic nodes (top-level and sub-topics count) and you can have up to 127 knowledgebases at your site. Queries can cross-reference these topics, but by default only the topics in the current selection will be used.

CONFIGURING THE SERVER

You will create a server configuration file that specifies server processing options, client processing options, and an optional security driver. Many other features can be implemented through the server as described in the following sample configuration file.

There is a significant performance hit if you enable the audit drivers so you may want to use this feature sporadically. Load balancing options are only available on UNIX versions of the Enterprise Server.

Example of the topicsvr.cfg File

```
#
# topicsvr.cfg
#
# Configuration file for SEARCH’97 Enterprise Server
#
#
$control: 1
#------------------------------------------------------------------
# The server section is used to control attributes and access features of TOPICSVR, as
well as most search-specific features as well.
#
server:
#--------------------------------------------------------------
# serverHome is used to tell the server what the working (current) directory is.
Regardless
# of what directory the TOPICSVR program resides in, or where it was run from, this
directory
# will be the starting location for where all collections, topic sets, and other
transactions
# will be looked for. The Verity default for serverHome is the directory where the server
was
# run from.
#
#/serverHome = .
#--------------------------------------------------------------

# maxUsers is used to limit the number of concurrent users who can access this instance
of
# TOPICSVR at one time. The Verity default for maxUsers is 16
#
/maxUsers = 16
#--------------------------------------------------------------
# maxProcess controls the number of processes that this instance of TOPICSVR is allowed
to
# start up. The more processes TOPICSVR can start, the better the performance, however
more
# system resources will be consumed. The Verity default for maxProcess is 1
#
/maxProcess = 1
#--------------------------------------------------------------
# minProcess controls the minimum number of processes that this instance of TOPICSVR is
# allowed to start up. The less processes TOPICSVR has started, the better the
performance,
# and more system resources will be available.  The Verity default for minProcess is 0
#
/minProcess = 1
#--------------------------------------------------------------
# overflowPercent sets the server load threshold where a new process will be started to
help
# the server service requests. The starting of a new process due to overflow requires
that
# the maxProcess limit has not already been reached. If it has been, no new process will
be
# started. For example, if maxProcess is set to 5 and there are currently two processes
# running, and another user logs in and runs a long search, TOPICSVR will start up
another
# process to service that third user. As the third user logs off, TOPICSVR will shut down
# that additional unused process. The Verity default for overflowPercent is 0
```

```
#
/overflowPercent = 0
#-----------------------------------------------------------------
# serverSpec controls the access method and port number that TOPICSVR uses to listen to
# connections from various clients.  The serverSpec consists of a number of values
separated
# by colons. The Enterprise Server Guide explains in detail about the various connection
# protocols and values associated with them.
#
# The Verity default for serverSpec is ONC:0x30909091:5, meaning TOPICSVR will attempt to
# talk to the server's portmapper to obtain port number 814,780,561 and will wait 5
seconds
# for a reply from the portmapper.  We use TCP in our center and the example shown below
is
# requesting port 5000 for the server.
#
/serverSpec = tcp:5000
#-----------------------------------------------------------------
# background determines whether or not to start the server as a background process. This
is
# similar to using an & on the UNIX command line when running TOPICSVR. The Verity
default
# for background is NO
#
/background = NO
#-----------------------------------------------------------------
# messageOfTheDay is a simple text string that is sent to each client that logs into
# TOPICSVR. Where the message is displayed depends on the client that is connecting to
# the server. In the case of TOPIC AGENTS, the message appears in the message window
# which may or may not be open. Keep in mind that sending this message from the server
# will not cause the client software to open the message window,  just to display it.
This
# message has no impact on the operation of the server or the client, but can be used to
# convey useful or helpful messages to the clients as they connect. # The Verity default
for
# messageOfTheDay is ""
#
/messageOfTheDay = "Welcome to Our Sports Server"
#-----------------------------------------------------------------
# maxDocs controls the maximum number of documents returned on the clients result list.
The
# purpose of setting maxDocs is two-fold. First of all, it keeps users from getting
# overwhelmed when a search is run that would normally return hundreds of thousands of
# documents that match the search criteria. Second, it prevents the system from using up
# valuable system resources gathering results information on documents the user will
# probably never use. The Verity default for maxDocs is 500
#
#/maxDocs = 500
{
#---------------------------------------------------------------------
# The session section is used to control session attributes and general server operation.
#
session_options:
#-----------------------------------------------------------------
# dataPath is used as an internal extension to the system's environment path. Instead of
# making additions directly to the system environment path, this line can be included to
do
# that.  The artificial path extensions effect only TOPICSVR, and remain in effect until
# TOPICSVR is shut down. When searching for files to open, TOPICSVR searches the data
path
# in the following order: first, TOPICSVR will look in the serverHome directory, then
along
# the environment path, then along the dataPath.
```

```
#
# To reference multiple directories in the dataPath, you must put a ":" between each path
# entry, for example: /dataPath = "data/demo:data/files:docs/files"
#
# If you are using an Adobe Acrobat collection, it is critical that the .pdf files can be
# found somewhere along the data path.  The Verity default for dataPath is ""
#
#/dataPath = c:\verity\s97ent\demos\data
#-----------------------------------------------------------------
# startupCollection is used to point to the collection(s) that are to be opened and made
# available to the clients for searching when TOPICSVR is started. startupCollection can
# point to a single collection or a group of collections referenced in a collection map
file
# (.clm). If you are pointing TOPICSVR at a single collection, you would set
# startupCollection to that collection's path: startupCollection = colls/news9610
#
/startupCollection = c:\verity\s97ent\demos\colls.clm
#-----------------------------------------------------------------
# vdkHome is used to point TOPICSVR to a directory named "common" which contains many
# important files that TOPICSVR needs in order To run. If TOPICSVR cannot find this
# directory, or does not have the permissions to read into that directory, it will be
unable
# to run, and the console will receive a message about not being able to open the message
# database. The "common" directory is copied to the system when the TOPICSVR installation
# program is run, and can be found under the product was installed into. If the
# common directory is moved for any reason, or can no longer be found along the system
# environment path, this value will need to be updated. The Verity default for vdkHome is
""
#
/vdkHome = common
#-----------------------------------------------------------------
# topicSet is used to reference a set of topics that is to be opened when the server is
# started and made available to the clients for search purposes. The discussion of what
# topics are is outside the scope of the text of this configuration file.
#
# The topicSet parameter is currently available, but will be eventually phased out, and
# emphasis will be put on using the startupKB parameter instead. Currently, topicSet and
# startupKB perform some of the same functions. The Verity default for topicSet is ""
#
# /topicSet = c:\is97\topics\news
#-----------------------------------------------------------------
# startupKB functions much like startupCollection and can be used to reference a single
# topic set, or a group of topic sets referenced by a map file. This is the primary
# advantage over the topicSet parameter, which can only open one topicSet per instance of
# TOPICSVR. Like startupCollection, to open a single topic set, you need to reference the
# location of the topic set, for example: /startupKB = kbases/finance
#
# To open a group of topic sets defined in a knowledgebase map file, point startupKB to
the
# map file: /startupKB = c:\verity\s97ent\demos\kbases.kbm. The Verity default for
startupKB
# is ""
#
/startupKB = c:\verity\s97ent\demos\kbases.kbm
#-----------------------------------------------------------------
# maxFiles controls the number of file pointers (handles) the client session can use. The
# default value for this member   depends on the operating system TOPICSVR is run from.
# Depending on what the client is doing, when the maxFiles setting   is reached, several
# different things can happen. In most cases,   the reaching of the maxFiles limit will
# generate a message and   probably stop whatever operation the server was performing for
# the client, for example stopping a search, stop returning  new or additional search
result
```

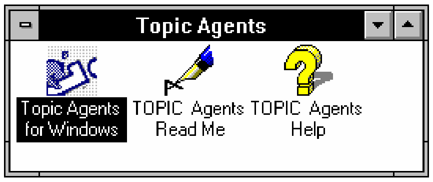
```
# information, etc. If the setting is reached, a message will be visible from within
Topic
# Agents   from within the Message Window, which may or may not be open.
#/maxFiles =
#-----------------------------------------------------------------
# maxPages determines the number of 1K blocks of memory that the instance of TOPICSVR
will
# attempt to allocate itself when it is started up. This is the setting for the internal
# disk cache, and it is this cache area where information such as user search results,
# search requests, search and collection statuses are stored, even if only temporarily.
The
# larger this cache, the   better TOPICSVR's performance will be, however this will be
# using up more of the host system's memory resources. This value can be set as low as 0,
# but the performance will be terrible.    Typically, settings for this value will be
between
# 128 and 512. The Verity default for maxPages is 128
/maxPages = 128
#-----------------------------------------------------------------
# threaded controls if TOPICSVR will be running as a multi-threaded server or a single-
# threaded (basically the same as a normal, non-threaded application) server. The
advantage
# of  running TOPICSVR in threaded mode is performance. By running  TOPICSVR threaded,
you
# are basically allowing it to perform several different tasks concurrently, scheduling
# different mini  tasks as time permits.  The best advice is to start running your server
in
# non-threaded  mode for a while to make sure it is working as you would expect it to,
then
# switch to threaded mode and run for a few days and see if any clients complain of
lockouts
# or other unexpected behavior. Fortunately, switching between   multi- and single-
threaded
# mode is as simple as changing this parameter.  The Verity default for threaded is YES
/threaded = yes
#-----------------------------------------------------------------
# accessQuestion is used to help narrow down the set of documents that are actually
# searchable from this server. For instance,  you can set up several servers, each of
which
# point to the same  set of 10 collections, but you want to limit one server to be able
to
# search only documents that contain the word "computer".  You would set up an access
# question that would look like:  /accessQuestion = "computer". From that point, any
client
# that connected to that TOPICSVR  would only be able to search across documents that
# contained the word computer, even though the 10 collections contain many other
documents
# that do not contain that word. Keep in mind that this  setting does not affect any
other
# instances of TOPICSVR that are  running against this set of collections.    The Verity
# default for accessQuestion is ""
#/accessQuestion = ""
#--------------------------------------------------------------------
# The drivers section is used to access various external components  and make them
available
# to TOPICSVR. In all cases, TOPICSVR does not require that *ANY* drivers be loaded in
order
# to function, but if the various drivers are not loaded, TOPICSVR will be unable to
# otherwise perform the function(s) associated with each driver.  Similarly, not all
drivers
# need to be loaded each time in order for TOPICSVR to run. The various drivers are not
# mutually inclusive or exclusive of each other.
drivers:
{
    #----------------------------------------------------------------
    # The audit drivers allow TOPICSVR to document various aspects of  its operation, from
    who
    # logged on, to what searches were run,  how long those searches took, and what results
    # were returned  just to name a few. The audit trail is in the form of a log file  that
    is
    # saved to the hard disk, as well as a series of messages  that will appear on the
    console
    # that TOPICSVR was run from. The audit driver is installed in two pieces. The first
    # piece, the LINK setting, tells TOPICSVR the name of the log file as  well as the types
    # of message to write to that file (-e -f).  The second line actually lodes the audit
    # driver code into memory  and registers it with TOPICSVR so it can perform the auditing
    # functions. Both the first and second line will vary from  hardware platform to hardware
    # platform.    TOPICSVR does not require an audit driver to be loaded in  order to
    # function, but without the loading of the audit driver,  it will not be able to generate
    # the audit trail log.    The Verity default for audit is: None audit: "LINK:0:SystemLog
    # -e -f nti31.log"  audit: DLL:_nti31/dll/samp_log.dll:audit
    # ----------------------------------------------------------------
    # The compression driver is used to reduce the amount of information that is actually
    sent
    # over the network between the client and the server. This does not actually reduce the
    # level of information sent back and forth, but rather stores it in a smaller, compressed
    # form that can then be de-compressed. The actual syntax for loading the compression
    driver
    # will vary from hardware platform to hardware platform. TOPICSVR does not require a
    # compression driver to be loaded in order to function, but without the loading of the
    # compression driver, it will not be able to send or receive compressed information. The
    # Verity default for compress is: None
    compress: DLL:_nti31\dll\drvr_rle.so:compress
    /name = rle
    #----------------------------------------------------------------
    # The encryption driver is used to protect the transmissions to and from the server. This
    # driver scrambles the data sent to a server from a client, then the server will
    unscramble
    # it. This is useful when the content of the queries or the content of the results are of
    a
    # sensitive or secret nature. The security this driver provides is not C2-class security,
    # but will keep its contents secure from the more casual browsers.  The actual syntax for
    # loading the encryption driver will vary from hardware platform to hardware platform.
    # TOPICSVR does not require an encryption driver to be loaded in order to function, but
    # without the loading of the encryption driver, it will not be able to send or receive
    # scrambled information. The Verity default for encryption is: None
    encrypt: DLL:_nti31\dll\drvr_xor.dll:encrypt
      /name = xor
    #----------------------------------------------------------------
    # The security driver is used to prevent unauthorized access of TOPICSVR. Basically, this
    # driver implements a name/password security mechanism that forces each potential user of
    # TOPICSVR to provide a user name and password before TOPICSVR will allow that client to
    # perform any searches. The actual syntax for loading the security driver will vary from
    # hardware platform to hardware platform. TOPICSVR does not require a security driver to
    be
    # loaded in order to function, but without the loading of the security driver, it will
    not
    # require a user name or password to be able to connect to TOPICSVR. The Verity default
    for
    # security is: None
    security: "LINK:0:VerityPassword"
    }
}
$$
```

STARTING TOPIC AGENTS

When you start Topic Agents, a special workspace configuration is checked to identify which document collections, topic knowledgebases, and personal agents should be opened for you. By default, Verity's demo database is selected for your use and the exercises in this guide have been written to take advantage of the information contained in this demo. Later, you will want to change your workspace configuration to point at your document collections, knowledgebases and agents.

Starting Topic Agents

Simply double-click on the Topic Agents icon to start your session

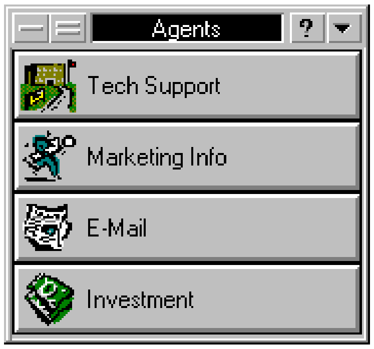


Connection Information

Double-click on the Topic Agents icon while holding down the Ctrl key to access the Connect dialog. You can also edit the command line under the Topic Agents Properties and add a space, then a -s to the end of the command line. This will cause the dialog to appear each time you start Topic Agents. See Appendix B: Connection Alternatives in for more information.

Opening an Agent

Agents available to you are presented on the Agent Manager. This button bar allows you to pick the agent you wish to open with just a click of the mouse.



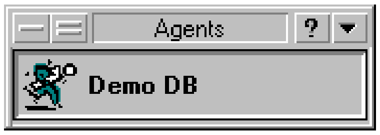
The agent keeps track of the collections you want to search, the topics, words or field values you want to search for, and even the presentation of results, including the field information to be displayed and the order and sorting of this information.

You can have many agents and each one can look at information in a unique way. The agent is designed to provide a shortcut for targeting just what you want to search for. Agents can be shared throughout your organization.

To help you get started, Verity ships a demo agent called Demo DB. This agent has a number of document collections you can search, along with a number of topics (pre-defined subjects) you can search with.

The Demo DB Agent

Simply click on the demo agent displayed in the Agent Manager

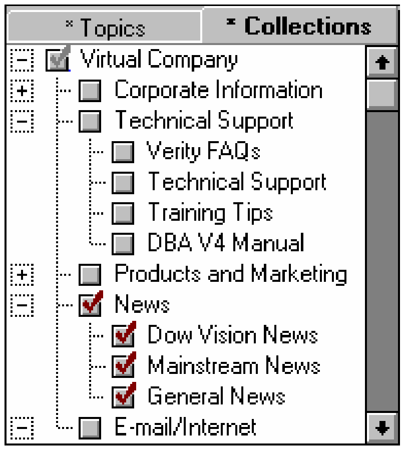


Selecting Collections

Collections represent groups of documents that have been processed so they are retrievable by Topic Agents. During processing, optimized indexes are automatically created to enable fast searching of words contained in the documents or attribute information about the documents like their titles or authors.

Available Collections

View the collections you may search by clicking on the Collections tab. Collections can be presented hierarchically, as shown in the demo. To see sub-collections, click the plus sign next to the collection name.



Enable/Disable Collections

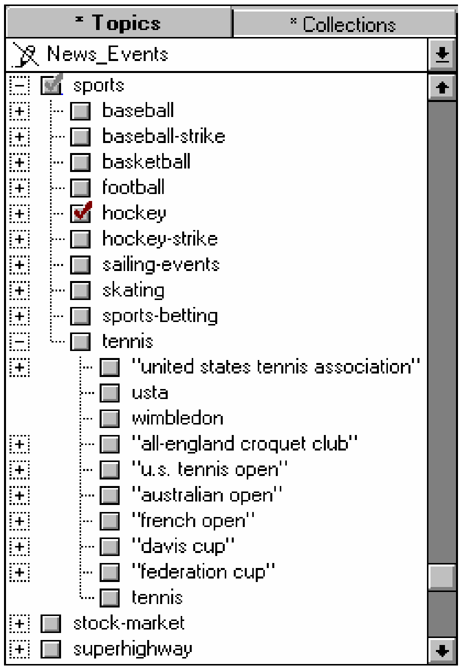
Collections are searchable when a checkmark is displayed in the gray box next to the collection name. Simply click the gray box until the checkmark appears to ready the collection. This is a toggle switch (on-off). If you do not wish to include a particular collection's documents in your search, you simply disable by clicking the gray box until the checkmark goes away.

Selecting Topics

Topics are pre-defined subjects designed to target a specific group of documents. For example, if you are interested in documents about sports in general, you can create a topic that includes all the typical words you would search for related to sports. Topics can be organized effectively into a group of sub-topics and each of these can be used independently or with other topics or search words. Topic Agents presents lists of Topic subjects organized in knowledgebases.

Available Topics

View topics you may search with by clicking on the Topics tab. Topics can be presented hierarchically, as they are in the demo. To see sub-topics, click the plus sign next to the topic name.

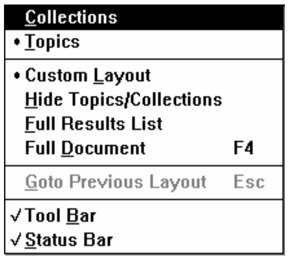


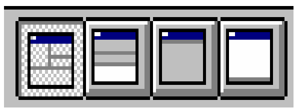
Enable/Disable Topics

Topics are enabled (on) when the checkmark is present, and are disabled (off) when the checkmark is absent.

Customizing Views

There are four views available to you when you are working with Topic Agents. The Custom View (first toolbar button) presents the most information on screen. The Hide Topics/Collections (second toolbar button) view provides more information for your query. The Full Results (third toolbar button) view provides more information for the results list. The Full Document view (fourth toolbar button) provides more information for reading the document.



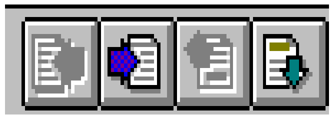


Changing your View

Click the appropriate toolbar button or select the view from the view menu.

Navigating Lists and Documents

You can move between documents on the list and between highlights in the document by using toolbar buttons or quick keys. These options are also available on the Document menu.



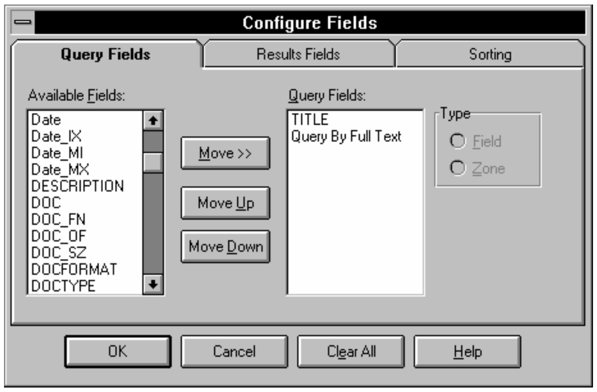
Previous Document	Click previous document toolbar button (first one) or F9
Next Document	Click next document toolbar button (second one) or F8
Previous Highlight	Click previous highlight toolbar button (third one) or Shift+F9
Next Highlight	Click next highlight toolbar button (fourth one) or Shift+F8
Search for Text	Use Ctrl-F to specify a string of text to search within the document you are reviewing

Searching by Fields

Sometimes it is more helpful to search information about your documents than the information contained in them. For instance, you may be looking for a particular type of document, or wish to search documents by title or within a date range. This kind of information is contained in fields and you can query against any of the fields defined for your Topic Collections.

Adding Query Fields

To add query field to the query window, open the Configure menu and choose Fields. Click on the field to add from the list of Available Fields, then click the Move button. You can specify the type (field or zone) if your DBA has defined zones for your collection. To remove a field, simply highlight it under Query Fields and move it back to Available Fields. Click OK.



Available Fields: Date, Date_IX, Date_MI, Date_MX, DESCRIPTION, DOC, DOC_FN, DOC_OF, DOC_SZ, DOCFORMAT, DOCTYPE

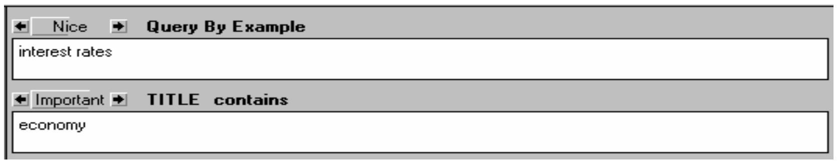
Query Fields: TITLE, Query By Full Text

Changing Field Order

You can easily change the order of fields appearing in Topic Agents. The current order is displayed under Query Fields. Highlight the field to re-order and click Move Up or Down.

Specifying Importance

You can indicate how important each part of your query is. Choices are Required, Important, Nice, Optional, Exclude.



interest rates

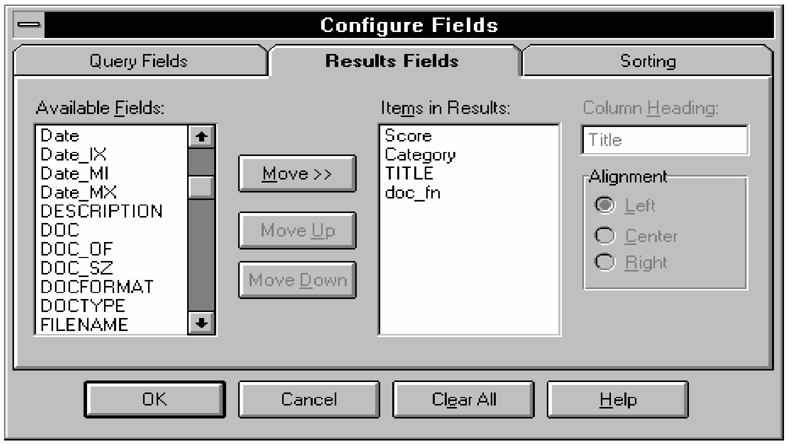
economy

Customizing Results

You can specify the fields to be displayed on your results list and the order of the fields. You can dynamically adjust the length to be given to any field and can move fields around on the fly.

Adding Fields to Results

To add field to your results list, open the Configure menu and choose Fields. Click on the second tab to move to the Results Fields. Highlight the field to add from the list of Available Fields, then click the Move button. You can specify the alignment and can provide a different column heading than the field name if desired. To remove a field, simply highlight it under Items in Results and move it back to Available Fields. Click OK.


Configure Fields

Available Fields:
Date
Date_IX
Date_MI
Date_MX
DESCRIPTION
DOC
DOC_OF
DOC_SZ
DOCFORMAT
DOCTYPE
FILENAME

Items in Results:
Score
Category
TITLE
doc_fn

Column Heading: Title

Alignment: [✓ Left] Center Right

Changing Field Order

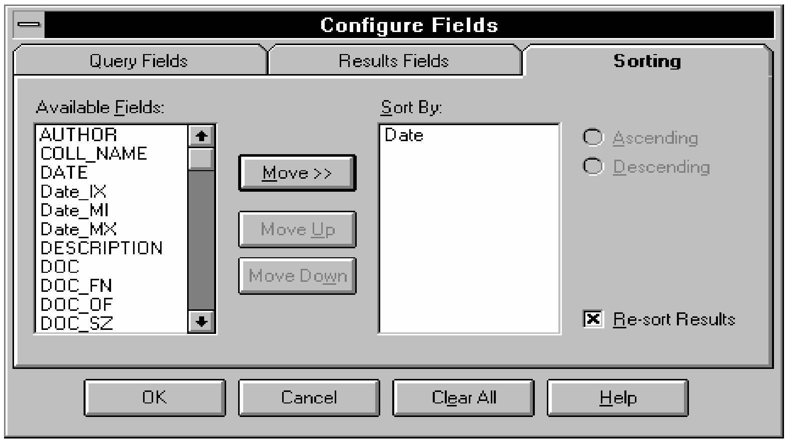
You can easily change the order of fields appearing in Topic Agents. The current order is displayed under Items in Results. Highlight the field you wish to re-order and click the Move Up or Move Down button.

Changing Field Display

You can change the width given to a particular field by pulling the vertical bar between the fields left or right to change the display. You can also grab the field name and move the entire column to a new position from the results list (similar to Move).

Customizing Sorting

You can specify how fields will be sorted on your results list and you can specify more than one field. Keep in mind that default sorting is by Score so that you are seeing the most relevant document first.



Sort By: Date

[✓ Re-sort Results]

Changing Sort Fields

To sort fields on your results list, open the Configure menu and choose Fields. Click on the third tab to move to Sorting. Highlight the field to add from the list of Available Fields, then click the Move button. You can specify whether sorting is to be in ascending or descending order. To remove a field, simply highlight it under Items in Results and move it back to Available Fields. Click OK.

Changing Sort Order

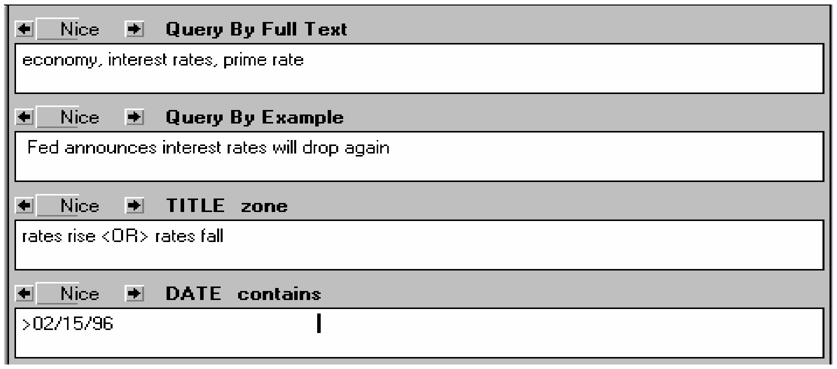
You can easily change the sort order of fields appearing in Topic Agents. The current order is displayed under Sort By. Highlight the field you wish to re-order and click the Move Up or Move Down button.

Query Types

There are four query types available within Topic Agents and each one looks at information in a unique way.

Query By Full Text	Searches based on words, phrases and topics you enter. This is the default query type.
Query By Example	Searches based on blocks of text you type or paste in, to locate documents *like* ones already found.
Field Query	Searches based on values you enter for query fields that have been defined for your collections.

Zone Query	Searches the text contained within particular regions of your documents, which have been defined as zones by your DBA.


Nice **Query By Full Text**
economy, interest rates, prime rate
Nice **Query By Example**
Fed announces interest rates will drop again
Nice **TITLE zone**
rates rise <OR> rates fall
Nice **DATE contains**
>02/15/96

CREATING PERSONAL AGENTS

It is very easy to create a new agent. If you are starting completely from scratch you will define four things.

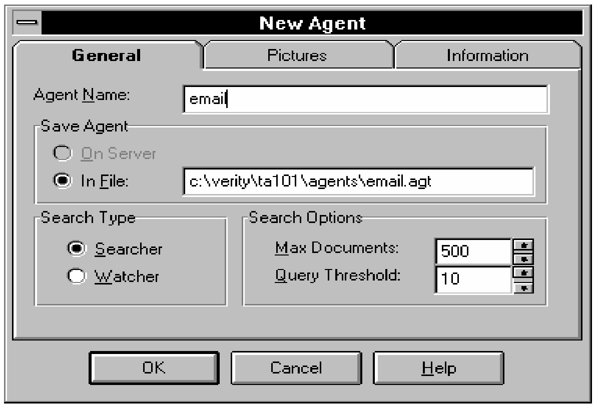
- Your agent's special properties (name, type, icon, etc.)
- Where your agent will search (collections)
- What your agent will look for (query type and topics, fields, or words)
- How your agent will present matching results to you (results list, view type)

Defining Agent Properties

Agent Properties are organized under three tabs: General, Pictures, Information. Each tab contains information about the particular agent you are creating.

General Properties

Name the agent a brief descriptive name (like email). *Save Agent in File* field shows the automatically created name of the actual file that contains your specifications. If you are saving an agent on the Server, you will not see the file name. *Search Type* refers to whether this agent will search continuously (Watcher) or only when you activate it (Searcher). *Max Docs* lets you set the maximum number of documents you want to see on the results and the *Query Threshold* refers to the minimum score each document much achieve in order to be qualified as a match.


New Agent

General | Pictures | Information

Agent Name: email

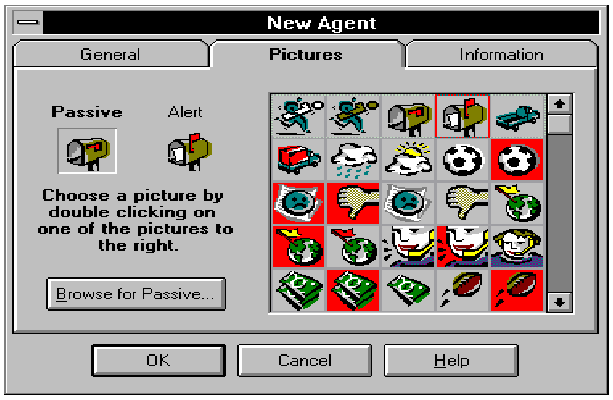
Save Agent
- ( ) On Server
- (●) In File: c:\verity\ta101\agents\email.agt

Search Type
- (●) Searcher
- ( ) Watcher

Search Options
- Max Documents: 500
- Query Threshold: 10

Pictures

The Picture properties allow you to select a picture, or icon, which will display when your agent is passive (no new information has arrived) or, if your agent is a watcher, active (new information is available).


New Agent

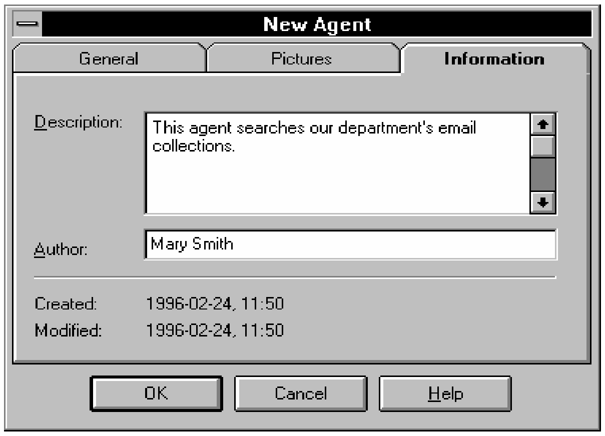
General | Pictures | Information

Passive | Alert

Choose a picture by double clicking on one of the pictures to the right.

Information

The Information properties allow you to describe your agent. It also display information about author and modification dates.


New Agent

General | Pictures | Information

Description: This agent searches our department's email collections.

Author: Mary Smith

Created: 1996-02-24, 11:50

Modified: 1996-02-24, 11:50

Lab Exercise - Adding an Enterprise Server

1. Open a Command Prompt window and move to the Server Admin directory under the install directory for the Enterprise Server (c:\verity\s97ent\tes222\admin). Create your own server configuration file by copying the topicsvr.nti configuration file:

   ```
   copy topicsvr.nti mysvr.cfg
   ```

   You could edit this file from here, making all the needed updates for port numbers and access to collections but you can also edit this file from within the Search'97 Enterprise Server Administration. For now, edit mysvr.cfg using wordpad.

2. Update the following sections in your configuration file, make sure to uncomment any that are commented out. Note that the collections and topics you will be using are coming from the Enterprise Server's demo area. Note also that we are adding an accessQuestion to limit documents to those that are ASCII text. The reason for this is that there is a bug in the current Topic Agents product related to viewing non-ASCII data. This is scheduled to be corrected in the next release.

   a. **`/serverSpec = TCP:54321`**
   b. **`/messageOfTheDay = "Welcome to My Personal Server" (or some other fun message)`**
   c. **`/startupCollection = c:\\verity\\s97ent\\demos\\colls.clm`**
   d. **`/startupKB = c:\\verity\\s97ent\\demos\\bases.kbm`**
   e. **`security: "LINK:0:VerityPassword c:\\verity\\s97ent\\tes222\\admin\\verity.pwd"`**

3. You may start the server using the Verity Service Administrator (Start → Programs → Verity → SEARCH'97 Enterprise Server Admin or at the Command Prompt by typing the following command line:

   ```
   c:\verity\s97ent\tes222\admin\topicsvr mysvr.cfg
   ```

   If you choose to start the server using Verity Service Administrator, from the Services menu, choose "Install Service." Name your service, using your student id: train1 and click the "Browse" button to locate your config file (mysvr.cfg). Highlight the config file and click the "Open" button. This allows you to edit this file from here.

4. Open a Command Prompt window and move to c:\verity\s97ent_nti31\bin. Here you will find the audit log file that is tracking actions. Enter the following command to review the contents of this file:

   ```
   type nti31.log | more
   ```

5. Connect to your server using Topic Agents. Click on the icon on the desktop and select the Remote tab for
   remote connection. Enter the following: (substituting your student number):

   ```
   topic Server Hostname:  tcp:train1:54321
   username:  guest
   password:  guest
   ```

6. When connected to the Enterprise Server, note the "Message of the Day" appears in the message window.

7. An Agent Manager also appears on your desktop. You currently have one agent created called "Demo DB", double-click on the "Demo DB" agent button to launch Topic Agents.

8. Spend a few minutes exploring the Topic Agents window. Both menu bar and tool bar are located at the top of the Agent window. Take note that the Single Magnifying Glass button is the "Search" button. In the left window pane contains a list of demo Collections and Topics. On the top right window pane is the Query box, right underneath is the Results List area, and below is the Document View window.

9. Review the Collections, check the box in front of a collection you want to enable for searching. Enter word(s) of your choice in the Query box, then press Enter or click the "Search" button on the tool bar. When documents appear in the Results List, double-click on a document to view it in the Document View window.

10. Practice with more searching. Enter words or phrases, and/or select a topic(s) for searching, review your results.

USING REGULAR EXPRESSIONS

Some style files require you to write pattern matching rules for capturing field information or for building word tokens in the word index. Regular expressions are used to select strings of text from your documents that will be used to populate fields in the Document Dataset VDB. When you write a regular expression you first define how data is to be located and then define what data is to be retrieved. Regular expressions are made by combining normal characters with a number of special characters. The character preceding the regular expression is the one that the regular expression refers to. For example, a period (.) means match any character and period asterisk (.*) means match any number of characters.

.	Matches any single character except newline. Spaces are also treated as characters. For example, p.p matches pep, pip, pcp, p p.
*****	Matches any number or none of the character immediately preceding. For example, be*t matches bt (no e), bet (one e), or beet (any number of e's).
?	Matches one or none of the character immediately preceding. For example, chapters? matches chapter (no s) or chapters (one s).
+	Matches one or more of the character immediately preceding. For example, Part .+ matches Part 1, Part 2, Part 11, or Part 1220.
$	Requires that the preceding character(s) must be at the end of a line.
^	Requires that the following character(s) must be at the start of a line.
\	Indicates to treat the following special character as an ordinary character. For example, \. matches an actual period instead of "any single character" and \$ matches an actual dollar sign instead of requiring the placement of the character to be at the end of the line. Use a double backward slash when you want to include a backward slash as itself.
[ ]	Matches any one of the characters within the brackets. For example, [AB] matches A or B.
[x-z]	Matches any one of the characters within the range starting with the first character and ending with the second character. For example, [0-9] looks for any number between 0 and 9.
[^]	Matches anything but the characters, which follow within the brackets. For example, [^a-z] looks for any character that is not a lowercase letter (this would find all uppercase letters, numbers, spaces, and non-alphanumeric characters).
[\\]	Most special characters lose their meaning within a bracket so you don't need to do anything special to use them as ordinary characters. However, three characters do require an escape: \ - ^.
(|)	Matches either of the strings before and after the | which are enclosed in parentheses. For example, Sub(j|ject) searches for both Subj and Subject.
\b	Defines a backspace.

\f	Defines a formfeed.
\n	Defines a newline.
\r	Defines a carriage return.
\t	Defines a tab.
\v	Defines a vertical tab.
{ }	Defines a symbol

< > Indicates to populate the field in the collection using only the substring enclosed within these angle brackets.

Lab Exercise - Practicing with Regular Expressions

1. What do each of the following expressions search for?

```
[0-9]

[mnt]

[r-z]

[A-C]

^Title:

\n

[0-9]$

DATE(:|LINE:)

[Tt]he

Volume [0-9]+

\*+

s?

[a-z]+\n

[a-z0-9]$
```

2. Create expressions looking for the following:

   Any uppercase letter

   Any uppercase or lowercase letter

   Any digit, one or more occurrences

   Any character followed by newline

   Zero or more occurrences of newline

   Any uppercase letter followed by a colon and one or more digits

   One or more occurrences of any lowercase letter at the end of a line

   A colon followed by any number (including none) of spaces or tabs

   A single digit followed by a period

Two digits followed by a backslash

3. For each of the following expressions, write down what they are searching for.

   a. `Title:.+`

   b. `[0-9]+/[0-9]+/[0-9]+$`

   c. `By[ \t]*.*`

   d. `Subjects?:.*`

   e. `[A-Za-z]+ [0-9]+, [0-9]+`

4. Match the expressions in the previous exercise to the following information:

a.	12/09/90	_____
b.	By L. Doctorow	_____
c.	Title: My Great Work	_____
d.	April 1, 1991	_____
e.	Subject: DBA Anxieties	_____

5. Create expressions to search for the following formats (don't try to match the specific information; assume that it will vary):

   a. Title: Eliminating Paperwork
      Title: Relativity Theory Yet Again

   b. TITLE Miscellaneous Office Stories
      Title Whistling Messengers

   c. January 12 1990
      April 25, 89

   d. 9/12/89
      10/30/90

   e. 2
      2.1
      2.1.2

6. Check your answers on the pages that follow.

REGULAR EXPRESSIONS ANSWERS

1. What do each of the following expressions search for?

[0-9]	**a number within the range of zero to nine**
[mnt]	**the lowercase letters m, n, or t**
[r-z]	**a lowercase letter within the range of r through z**
[A-C]	**an uppercase letter within the range of A through C**
^ Title:	**Title: at the beginning of a line**
\n	**a newline**
[0-9]$	**a number within the range of zero to nine found just before the end of a line**
DATE(:|LINE:)	**the words DATE: or DATELINE**
[Tt]he	**the words The or the**
Volume [0-9]+	**the word Volume followed by any number (but at least one) of digits (Volume 1, Volume 126)**
*+	**any number (but at least one) of asterisks (***)**
s?	**one or none of the letter s**
[a-z]+\n	**any number (but at least one) of lowercase letters up to the end of a line**
[a-z0-9]$	**one lowercase letter or digit preceding the end of the line**

2. Create expressions looking for the following:

Any uppercase letter **[A-Z]**
Any uppercase or lowercase letter **[A-Za-z]**
Any digit, one or more occurrences **[0-9]+**
Any character followed by newline **.\n**
Zero or more occurrences of newline **\n***
Any uppercase letter followed by a colon and one or more digits **[A-Z]:[0-9]+**
One or more occurrences of any lowercase letter at the end of a line **[a-z]+$**
A colon followed by zero (none) or more spaces or zero (none) or more tabs **:[\t]***
A single digit followed by a period **[0-9]\.**
Two digits followed by a backslash **[0-9][0-9]**\\

3. For each of the following expressions, write down what they are searching for.

a. "Title:.+"
The string Title: followed by any number of characters.

b. "[0-9]+/[0-9]+/[0-9]+$"
one or more digits followed by a slash, followed by one or more digits, followed by a slash,
followed by one or more digits, preceding the end of the line. This would likely be a date: 09/15/96 at the end of the line.

c. "By[\t]*.*"
The word By followed by any number of or no spaces or tabs, followed by any number of or no characters.

d. "Subjects?:.*"
The string Subject: or Subjects: followed by any number of any character.

e. "[A-Za-z]+ [0-9]+, [0-9]+"
Any number (but at least one) of uppercase or lowercase letters followed by a space then one or more digits followed by a comma and a space, then one or more digits. This would likely be a date: January 17, 1992 or Feb 1, 1950.

4. Match the expressions in the previous exercise to the following information:

12/09/90	**B**
By L. Doctorow	**C**
Title: My Great Work	**A**
April 1, 1991	**E**
Subject: DBA Anxieties	**D**

5. Create expressions to search for the following formats (don't try to match the specific information; assume that it will vary):

a. Title: Eliminating Paperwork
Title: Relativity Theory Yet Again
"Title:[\t]*.*"

b. TITLE Miscellaneous Office Stories
Title Whistling Messengers
"(TITLE|Title)[\t]*.*"

c. January 12 1990
April 25, 89
"[A-Z][a-z]+ [0-9]+,? [0-9]+"

d. 9/12/89
10/30/90
"[0-9]+/[0-9]+/[0-9]+"

e. 2
2.1
2.1.2
"[0-9]\.?[0-9]?\.?[0-9]?"

FIELD DEFINITION: STYLE.TDE

The style.tde file may be a key component for your collections when the information you want to capture for fields, is actually contained within the document. Among other things, the style.tde file contains the rules to follow for populating fields in the document dataset when there a lack of tagging or structure prevent the automatic filtering of field information. Only those fields that have been defined in the style.ddd (through the $include to the style.ufl) can be populated when the utilities are run to create the collection.

The Datamap Statement

The datamap statement identifies the section of the style.tde that will deal with the parsing of documents. This statement specifies how data is to be mapped out in the collection. The datamap statement may have three modifiers:

```
/docsep = "<<End>>\n"
/filter = "auto"
/system = "system call"
```

The Field Statement

The field keyword identifies the rules for parsing data for a specified field name. A field keyword must be present for each field that you want to appear in the collection and it must be an exact match for the corresponding field identified in the style.ddd file.

```
field: TITLE PATTERN "Title: <.*>"
```

Modifiers for field include:

```
/required         field must be in document - yes or no (default is no)
/which            how to behave when multiple instances are found
                  (1,#,LAST,ALL)
/string-before    insert string before the field
/string-between   insert string between field values when many values for the
                  field exist
/string-after     insert string after the field
/default          value to be assigned if Topic does not find the value
/alsowrite        allows you to store a parsed value in two fields (one for the
                  original field defined by the FIELD statement and one for the
                  /alsowrite field
```

The Define Statement

The define statement identifies a symbol that is associated with a pattern written as a regular expression. This symbol can be included in another regular expression specified in the style.tde file.

```
define: CHAR "[A-Za-z]+"
define: SPACE "[ \t]"
```

The Dispatch Statement

The dispatch keyword specifies the rules for populating the document dispatch field as defined in the style.ddd file. Verity products display document text based on information provided by the dispatch keyword about where the document starts and ends. The document start and end can be identified by a line number or can be written as a regular expression. The default is to dispatch the entire document.

```
dispatch:   DOC
```

Dispatch Modifiers

/required	field must be in document - yes or no (default = no)
/start-line	line on which document begins (default = 1)
/start-pattern	pattern identified by regular expression for starting point
/end-line	line on which document ends (default = end of file)
/end-pattern	pattern identified by regular expression for ending point
/inclusive	identifies if start/end patterns are included in dispatch field (yes or no (default = no)

The Pre-process Section

This section is responsible for parsing and capturing field values in your documents.

While the parsing of field values is important, this section is also capable of performing other functionality. This section can also assign values directly to fields, perform system calls which can execute any command or script file that can normally be executed from your operating system's command line, and can write status or informational messages to either a log file.

The Form Section

The form section of the style.tde file configures what Topic Document Entry users see on their worklist. This product provides graphical collection building features and the form is important as it allows users to enter, edit, or validate field values for each document before documents are added to the collection. The only time the form section is used is when you are using the TDE program.

The Post-processing Section

This last section in the STYLE.TDE file is used for final cleanup and processing of each document before the work is submitted to the collection. Much like the pre-processing section, this section can assign field values, perform system commands, and write messages to a log file.

There are several uses for the post-processing section. The first is transferring document files from one location to another after they have been validated. Documents may be placed in a common directory as they are created or downloaded or scanned. Another good purpose for the post-processing section is to hide or alter field values after the user has validated the field. For instance, you may have a field called author. While the documents may show an author name of Author #1, your style file knows that the real author name for Author #1 should be K. Vadasz. In this case, pre-processing might capture a default value, which could be reassigned in post-processing.

SAMPLE STYLE.TDE FILE

```
$control:1
tde:
{
    pre-process:
        /relative-path = yes
    {
```

```
                datamap:
                    /filter = auto
                {
                    field: Subject PATTERN "Subject[ \t]*<.*>"
                        /default = “Needs Subject”
                    dispatch: DOC
                }
                log: vde-log "Finished pre-processing -- $DOC_FN"
                message: "Finished pre-processing -- $DOC_FN"
            }
            form:
            {
                field: "Subject"
                    /width = 60
                    /height = 3
                {
                     validate: not null
                }
            }
            post-process:
            {
                label: DonePOST
                log: vde-log "Finished Post-Processing -- $DOC_FN"
            }
        }
$$
```

LAB EXERCISE - EXPLORING STYLE FILES

1. Using the TDEASCII style files as a starting template, create a new set of style files called “mystyle” In the c:\is97\styles directory. It is easiest to do this from Windows. (Copy the TDEASCII style directory and rename the copy).

2. Edit the style.ddd to add a new field called “editor.” Make this a variable width field and place it toward the bottom of the file with the rest of the user-defined fields.

```
…
varwidth:    Filetime     dv4    # System Date and Time on File
varwidth:    Editor       dv5    # Document Editor Field
```

3. Edit the style.tde file. Note the three areas: pre-processing, form and post-processing. The form section is only used when you have the Topic Document Entry product. This product allows you to view values, validate data entry, etc. as documents are added to the collection. The only section you will use is pre-process.

4. Review the information under this section noting the following:

   a. The collection is to be built using relative path
   b. The document separator for multi-document files is set
   c. The parsing rules for existing fields show variations and, in some cases default values to be used if a matching pattern is not found within the document
   d. Additional processing is happening related to the date. In this case, if no match is found on the date pattern, a default of 1/1/80 will be assigned. If the date is 1/1/80, the indexer is to look up the system date and time for the file, and reassign this value. In this way, even if the parsing doesn’t produce a good date, the reassignment to the system date and time will get you closer to a true date.

5. Add the editor field. Write the regular expression to capture this field from the document based on the example below and add a default value for editor using your first and last name. Be sure to write the expression so it captures Editor, editor, and EDITOR.

```
Title:       A Tale of Styles
Author:      Cindy Johnson
Editor:      Kathy Vadasz
Date:        June 20, 1997
Source:      SEARCH’97 Class

Style files allow you to configure the collection.  Options you select
have an impact on the content and operation of the collections you build.
```

6. Create a new document directory (c:\is97\docs\doc8) and create 2 documents in the format shown above. In the first document, build a value for each field as shown. In the second document leave out some of the information to allow the defaults to be used (as shown below).

```
Title:       Collection Building
Author:      Cathy Ferrario
Source:      SEARCH’97 Enterprise Program

Collection building is lots of fun!
```

7. Build a new collection, ensuring the following:

   a. The collection is added to the main collection directory
   b. You use the new style files you just created
   c. You use the documents you created
   d. You create a batch file in the tools directory with an associated log file
   e. You test the collection using rc-vdk to search the words you entered and browse to see the values captured in your fields.

8. For this same collection, edit the style.dft and add the Editor field information (with a label like the other fields) just before the source field. This is the one style file that can be changed dynamically. Run rc-vdk again, to view the document and you will notice that the new field information displays in the "virtual document" you are creating.

9. Create a stop list (style.stp) in your style directory – as shown below:

```
the
it
your last name
```

10. Delete the collection you just created (easiest from Windows - just delete the whole directory) and then rebuild the collection using your batch file. This time, words will be stopped, simply because you put that style file in the style directory.

Use rc-vdk to try to search for the words that have been stopped. As you read through documents you see the words are still there because stopping words does not affect the actual document. They are simply not in the word index. Also, keep in mind that the word index is case sensitive, and you have only stopped the specific case variation you see in the file. If you stop "the" but not "The" and then search on "the" - you will find documents. Why? Because the search engine by default searches case-insensitive. But, if you instead query on <case>"the" - no results will be found.