Verity Intelligent Classification Guide

Version 6.1

November 30, 2005
Part Number DM0707

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

Verity Benelux BV
Coltbaan 31
3439 NG Nieuwegein
The Netherlands

Contents

Figures, Tables, and Listings

Preface

This guide is for administrators and developers for Verity Intelligent Classifier. Many of the administration functions described in this guide can be performed by developers, information scientists, and librarians, depending on the structure of your enterprise.

This guide is intended for users with a knowledge of the operating system and a basic knowledge of K2 classification applications. See *Verity K2 Getting Started Guide* for information about the K2 system.

This preface contains the following sections:

Using This Book

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

Version

The information in this book is current as of K2 Enterprise version 6.1. The content was last modified November 30, 2005. Corrections or updates to this information may be available through the Verity Customer Support site; see "Verity Technical Support" on page 20.

Organization of This Book

This book includes the following parts, chapters, and appendixes:

- Chapter 1, "Information Classification" describes Content Classification solutions in general terms.
- Chapter 2, "Intelligent Classifier Concepts" describes various Verity Intelligent Classifier concepts and in particular, understanding and working with topics, categories, workspaces and knowledge trees.
- Chapter 3, "Installing Intelligent Classifier" describes content classification solutions in general terms.

- Part I, "Building Taxonomies"
  - Chapter 4, "Creating Categories" describes the various ways you can create categories from document collections.
  - Chapter 5, "Thematic Mapping" describes *Thematic mapping*, which automatically extracts concepts and concept hierarchy from document collections.
- Part II, "Defining Categories and Creating Topics"
  - Chapter 6, "Using the Topic Editor" describes how to build topic sets using Intelligent Classifier.
  - Chapter 7, "Creating Topics Using Logistic Regression Classifier" describes LRC, an enhanced state-of-the-art machine-learning algorithm that can automatically create a Verity topic from a set of positive and negative exemplary documents. The `mklrc` command-line tool is also discussed.

- Chapter 8, "Creating Topics Using Hybrid Approaches" describes how to combine topic creation methods. Combining two or more methods can improve the quality of the resulting topics or reduce the amount of effort required.

- Part III, "Populating Taxonomies"
  - Chapter 9, "Parametric Index Creation" describes how to create and populate Parametric Indexes from a taxonomy or VDK field in Intelligent Classifier.
  - Chapter 10, "Building Knowledge Trees From the Command Line" describes how to build and maintain knowledge trees for use in a K2 Search System with the Knowledge Tree Management tool (`ktmgr`), the Taxonomy Management tool (`taxmgr`), and the Topic Set Conversion tool (`top2tax`).
- Part IV, "Deploying"
  - Chapter 11, "Testing Knowledge Trees and Parametric Indexes" describes various command-line tools for testing Knowledge Trees and Parametric Indexes. These tools include the Knowledge Tree Search tool (`ktsrch`) the Optimized Knowledge Tree Search tool (`qsrch`), and the Taxonomy and Parametric Index Search and Browse tool (`rcodk`).
  - Chapter 12, "Deploying Knowledge Trees, Parametric Indexes, and Topic Sets" describes how to deploy the knowledge trees, parametric indexes and topic sets built with Intelligent Classifier to end users for browsing categories.
- Appendixes
  - Appendix A, "Setting Style Options" describes setting Style Options, setting Intelligent Classifier Options and using Locales.
  - Appendix B, "Using Taxonomy Definition Files" describes building an outline file to create topic sets using `mktopics` and Intelligent Classifier. The `.otl` file is required to make topics with `mktopics`, but is optional for use with Intelligent Classifier. Information includes options and command syntax for a topic outline (.otl) file.
  - Appendix C, "Using Configuration Files" describes automatically-created files that you can use in other Verity applications. These files include Knowledge Base Map (KBM) files, TAX (taxonomy) files, and BIF files. Collection Map (CLM) files, which are only supported for knowledge trees, are also described in this appendix.
  - Appendix D, "Valid Verity Query Language Operators and Modifiers" lists VQL operators and modifiers available in Intelligent Classifier.
  - Appendix E, "Getting Started with the Quick Start Guide" describes the keystrokes and their corresponding actions for various commands within Intelligent Classifier.
  - Glossary

Stylistic Conventions

The following stylistic conventions are used in this book.

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

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

Convention	Usage
`[ optional ]`	Brackets describe optional syntax, as in `[ -create ]` to specify a non-required option.
|	Bars indicate "either | or" choices, as in `[ option1 ] | [ option2 ]` In this example, you must choose between `option1` and `option2`.

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

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

Related Documentation

The following guides provide more details on Verity products:

- *Verity K2 Getting Started Guide*
- *Verity K2 Dashboard Administrator Guide*
- *Verity Query Language and Topic Guide*
- *Verity Organization Developer's Kit Programming Guide*

Verity Technical Support

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

Telephone: (403) 294-1107

Fax: (403) 750-4100

Email: tech-support@verity.com

Web: http://www.verity.com

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

https://customers.verity.com

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

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

1

Information Classification

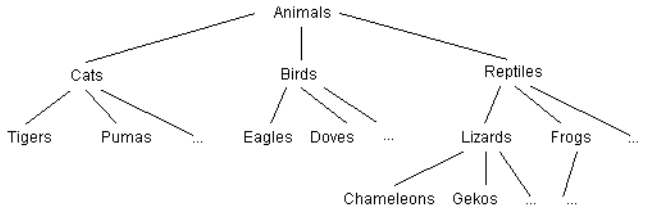
The following sections briefly describe information classification techniques and Verity's solutions in this area. If you are already familiar with these techniques and Verity's approach to information classification, you do not need to read these sections; however, you may find them useful as a review.

The sections are as follows:

Overview

Classifying information into categories and subcategories allows your users to locate individual files by drilling down through the concepts represented by each category until they find relevant documents. *Classification* organizes the documents in your collections into content hierarchies, called taxonomies. A *taxonomy* is a hierarchal organization of information by categories, such as kind of animal, as shown in Figure 1-1.

Figure 1-1 Sample Taxonomy Containing Animal Categories



As the number of documents used by an organization increases, the need to classify them into an intuitive and meaningful hierarchy becomes imperative. A major issue for many organizations is how to create such a hierarchy without examining each document, especially when the categories themselves have not already been explicitly identified. This issue becomes acute when there are thousands or millions of documents to classify.

Verity's classification solution enables a flexible approach that can combine subject matter (domain) expertise with automatic classification for documents in all popular formats. This solution enables you to organize your information assets into categories, making them easy for users to browse.

Verity's classification infrastructure can be implemented by using:

- Verity Intelligent Classifier, a graphical user interface-based tool
- a set of command line tools provided by Verity
- the Verity Organization Developer's Kit (ODK), which enables you to embed the infrastructure in your application programatically

Creating, Populating, and Using Taxonomies

There following four basic stages for creating, populating and using taxonomies are explained in the following sections.

- Building the Taxonomy
- Defining Categories
- Populating the Taxonomy With Documents
- Using the Taxonomy

Note In some situations, the first two stages, or even the first three stages, may be combined. These situations are also identified in the following sections.

Building the Taxonomy

By organizing information into a hierarchy of categories, a taxonomy, you can define a view of your content. The result of this stage is a skeletal navigation structure of categories, each with a name. You can have a category "Animal" with child categories labeled "Cat" and "Bird." These child categories can themselves have descendants, and so on. Several techniques exist for building a taxonomy:

- Using human domain experts

  Taxonomies built by domain experts are quite common. A domain expert typically builds a skeleton taxonomy and assigns names to its categories.

- Importing from an existing hierarchy

  This technique allows users to create a taxonomy by extracting the implicit hierarchies from existing URLs or file system hierarchies, or hierarchies defined in metadata such as the Dewey decimal number in a library catalog, and mirror them in a taxonomy.

- Using an industry-specific taxonomy

  Vendors such as Lexis-Nexis provide taxonomies for particular industry segments.

- Using concept mapping, extraction, and naming

  Thematic mapping automatically extracts key concepts contained in a set of documents and organizes them into a hierarchy called a *concept tree*. The Verity thematic mapping engine analyzes your documents and groups together concepts that recur throughout

the *corpus* into categories. It further creates a taxonomy structure for these concepts, all automatically. Automatic naming generates labels for these categories using linguistic analysis.

Defining Categories

After the taxonomy has been created, the next step is to attach category definitions to the taxonomy so that the taxonomy can be populated with documents. Each category definition consists of a mathematical rule against which each document can be evaluated for membership in that category. A very simple (and incomplete) rule for the category Animal could be defined as follows:

"If a document contains the word "paw" or "hoof," include the document in the Animal category."

As with creating taxonomies, there are several techniques for building rules that define categories. They include the following:

- Expert-defined rules

  A domain expert can use Verity's powerful topics, which are described in the section "What is a Topic?," to define a rule for each category. These rules are sometimes called business rules because the domain expert typically tailors them so that they are relevant to a specific business or business function.

- Importing from an existing hierarchy

  This technique allows users to define categories by extracting the implicit hierarchies from existing URLs or file system hierarchies, or hierarchies defined in metadata such as the Dewey decimal number in a library catalog. This technique is useful if documents' membership in categories corresponds to an implicit hierarchy, such as a file system or URL hierarchy. In this situation, the first two stages, building the taxonomy and defining categories, are combined into one stage.

- Using industry "standard" categories

  A standards body or independent vendor can create category definition rules for an industry's vertical taxonomies. This technique is closely associated with using industry-specific taxonomies. Using industry-standard taxonomies with standard categories can be combined, creating another situation in which the stages of building a taxonomy and defining categories can be combined.

- Automatic category creation

  An automatic classification system that creates categories is fed positive and negative example documents, called training documents, that respectively denote membership

in or exclusion from each category. The system "learns" from these training documents and creates a defining rule for each category. Verity's highly accurate Logistic Regression Classifier (LRC), which is based on state-of-the-art machine learning technology, implements automatic category creation.

With the Verity classification infrastructure, you can implement any combination of these methods. Best results typically are achieved by combining automatic capabilities with domain expertise.

Populating the Taxonomy With Documents

After the taxonomy is built and a rule bound to each of its categories, you can populate the categories with your documents. You can populate the taxonomy either manually or automatically, or both ways, as follows:

- Custom

  For each document, an expert determines the categories that should be populated and then explicitly populates those categories in the taxonomy with the document.

- Automatic

  The system evaluates each document against the rule for each category and assigns the document to the appropriate categories in the taxonomy.

You can combine both methods as you choose. Documents can be assigned to categories based on defined rules, or you can manually override those rules and place a document in a different category or in additional categories.

Note The way in which a document is assigned to a category can be implied, in some cases, by the import techniques for taxonomy building and for category definition; for example, the document's URL or path name might be used assign a category to the document when the taxonomy is created or a category is defined.

Using the Taxonomy

After building the taxonomy, defining categories, and populating the taxonomy, your taxonomy is ready to use. Studies show that with a well-implemented taxonomy, users perform a balanced combination of search and browsing to locate documents. The search could be issued at the top level to filter the categories in which matching documents exist, or contained within a subset of the taxonomy.

To accommodate the different ways that various groups within your enterprise use information, you can create multiple taxonomies to organize content in ways that make the most sense for each group. For example, separate taxonomies can be created for the sales, marketing, human resources, and engineering departments. This puts information into the context of your overall business model, and adds a valuable dimension to the content discovery process.

Relational Taxonomies

With relational taxonomies you can simultaneously navigate multiple taxonomies. You can drill down and navigate in the manner most intuitive to you, including simultaneously walking down multiple taxonomies at the same time. This feature is implemented through Verity's Parametric Search component. See the *Verity K2 Parametric Developer Guide* for more information.

Profiling

The earlier discussion of information classification assumes that batches of documents have been indexed into a Verity collection before being classified. In many situations, however, documents arrive incrementally over time and must be classified upon arrival, possibly one at a time. Consider an example in which each email message arriving into a customer response system may have to be routed to one of several service specialists. The content of the email is the basis on which the K2 System must decide which specialist to route the message to.

The Verity Profiler facilitates this kind of classification. A set of *profiles*, such as one for each specialist, is used to classify the document; each profile is expressed as a Verity topic. The set of profiles is compiled into a *profilenet*, an internal representation that optimizes the efficiency of the Verity Profiler. Arriving documents are evaluated against the profilenet and can be processed based on whether they match one or more profiles.

For further information about the Verity Profiler, see the *Verity Profiler Programming Guide* or the *Verity K2 Profiler Programming Guide*.

2

Intelligent Classifier Concepts

Intelligent Classifier is a tool for creating, viewing, editing, and testing Verity topics, taxonomies, knowledge trees, and parametric indexes. This chapter covers the following aspects of category search.

Topics and Topic Sets

The previous chapter gave a high-level overview of the various stages of building and populating a taxonomy. You can use the taxonomy to create a parametric tree for use by an end user.

This section introduces a fundamental building block in Verity's classification infrastructure, known as a topic. Topics play a central role in the category definition stage, which is described in "Defining Categories." It is very important to understand the role of topics in Verity's classification technology. The following sections define a topic and explain the relationship between topics, topic sets, and categories. For additional information about topics, see the *Verity Query Language and Topic Guide*.

What is a Topic?

A *topic* is a stored query expression that is written in the Verity Query Language (VQL). A topic models a concept of interest, which is used as the definition for a category. When a topic is evaluated against a set of documents, the Verity search engine identifies the subset of the documents that match the concept that the topic represents.

Consider the following scenario:

- You can use the VQL expression `GM <OR> Ford <OR> Chrysler` to model the concept "North American car manufacturers." When this expression is evaluated on a set of newspaper articles, the Verity search engine selects all articles that mention GM, Ford, or Chrysler as matching the concept "North American car manufacturers."
- Topics can be combined using VQL operators to create more complex topic definitions. For example, you might combine the concept "North American car manufacturers" with "European car manufacturers" (another VQL expression). By combining these topics and applying `<NOT>` to the concepts, you could perhaps create a new topic definition corresponding to the concept "Asian car manufacturers." (This definition assumes no South American or Australian car manufacturers.)
- You can also use sophisticated non-Boolean VQL operators.

Note In a realistic topic, you would use more powerful VQL operators, such as the `<ACCRUE>` operator instead of `<OR>`. VQL also has a role in advanced searching; however, a discussion of VQL is beyond the scope of this book. For information about VQL, see the *Verity Query Language and Topic Guide*.

Using individual topics or combining topics you can create *category definition* rules that are used to decide whether a document belongs to the category. There are several techniques for constructing topics, ranging from domain expertise to the use of automated machine learning techniques. Topics can be combined regardless of how they have been created. One advantage of combining topics is that it allows a gradual build-up in such a way that basic topics can be shared between multiple higher-level topics.

Relationship Between Topics and Topic Sets

A *topic set* is a grouping of stored queries or topics that have been compiled for use by a Verity-enabled application. Because a topic can be used to define a category, a topic set contains one or more topics used for classifying documents in a collection.

Categories and Taxonomies

This section describes how Intelligent Classifier interface defines, configures, and stores categories and taxonomies.

Categories and Properties

Each category must have a unique ID that identifies it within the taxonomy. You use the *category ID* to specify categories for browsing and scoped searching. You can also use it to specify categories assigned to documents when populating the Knowledge Tree, such as in document metatags, collection fields, or manual submissions to the Knowledge Tree. In the library science sense, the set of category IDs represents a controlled vocabulary for the categorization and indexing of documents.

Categories can also have one or more unique aliases that you can use like the ID to reference the category in all parts of the system. Often the ID is system-generated or otherwise cryptic, so you might want to provide a "user-friendly" *category alias* that users can explicitly reference.

You can also define properties for the category. Two important properties used by the system are Name and Description. The category name appears in the taxonomy view and provides a short, meaningful label for the category. The optional category description

provides a longer description of the category that differentiates it from other categories with the same name. Remember that although several categories can have the same name, each category has a unique ID.

In addition to these system properties, categories can be assigned other custom properties that are used by customized navigation interfaces.

Structure

All categories must have at least one parent category and can have none, one or more subcategories.

Every taxonomy has a special category called a Root from which all other categories available to end users descend. There can only be one Root category in a taxonomy. Each taxonomy has a second special category called Unused that contains all categories that are not available to end users. There can be only one Unused category in a taxonomy. The only difference between categories descending from Root and Unused is their availability to end users.

Documents can be assigned to categories at any level in the taxonomy. The category ID itself uniquely identifies the category in the taxonomy. This means that you can move a category from one part of a taxonomy to another without having to change all the places where that category is referenced. With this context-independent addressing scheme for categories, you can easily move a category without having to modify the category assignments for all of the documents in the category. When you assign documents to categories or refer to a category, you only need the category ID, you do not need a path.

Sometimes a category naturally belongs in multiple places in a taxonomy. For example, a category corresponding to a specific product on a company's intranet might belong underneath both the Research and Development category and the Marketing category. Categories can have more than one parent. When categories have multiple parents, the taxonomy is actually a directed graph rather than a tree structure.

When a category has multiple parents, a separate value for a category property can be specified in the context of each parent. These are known as local properties and can be thought of as properties on the link between the category and a parent. For example, a category might have a different name in the context of one parent than it does in the context of another parent.

When you move a single category from the *Root* category to the *Unused* category, all references to it are broken. This is necessary to prevent end users from possibly seeing a hidden category as a cross reference. A category can have multiple parents as long as parents do not exist in both the Root and Unused categories. When you move a category

and all of its parents from Unused to the Root, the links to all parents will remain intact. The same is true when moving categories from Root to Unused, as long as the parents are moved with the category.

Every taxonomy has three reserved categories, as described in the following list.

- Root—All categories descending from Root are available to end users for browsing and searching.
- Unused—All categories descending from Unused remain hidden from end users.
- Orphans—When you create a category without specifying its parent, it is automatically placed under the Orphans category.

There can be only one Root category, Unused category, and Orphans category in a taxonomy.

Taxonomy Definition

You create and edit categories in a Verity taxonomy file, in either text or XML format. A `.tax` file (referred to as a TAX file) stores a taxonomy in text format, while an `.xml` file stores the taxonomy in XML format. This allows importing one set of category definitions into another.

The taxonomy file is a useful transport and archival mechanism for taxonomies that enables taxonomies to be easily defined by processes or scripts. You can export a taxonomy database to a TAX or XML file for archival or external editing. See "Using Taxonomy Definition Files" for more information.

Taxonomy Index

In addition to storing the taxonomy definition, a taxonomy database stores a searchable collection of categories, full-text-indexed by category name and description. This enables the browse server to return a list of categories whose name and/or description match a user query, a popular feature found on some Internet navigation services. The `idxstyle/style.ufl` file defines the schema for the taxonomy index and controls which category properties are stored in the collection.

Understanding Knowledge Trees

Verity Knowledge Trees are used by K2 to organize your data into browsable categories. A Knowledge Tree consists of a taxonomy plus documents that are assigned to the categories in the taxonomy and other information about the documents. It is created from one or more taxonomies that you develop in Intelligent Classifier, or with command-line tools. You can use the Intelligent Classifier to create, edit, populate, and test taxonomies and Knowledge Trees.

To accommodate the different ways that various groups within your enterprise use information, you can create multiple taxonomies to organize content in ways that make the most sense for each group. For example, separate taxonomies can be created for the sales, marketing, human resources, and engineering departments. This puts information into the context of your overall business model, and adds a valuable dimension to the content discovery process. A taxonomy can be used by more than one Knowledge Tree.

Understanding Workspaces

As well as enabling you to work with Knowledge Trees, Intelligent Classifier enables you to work with *workspaces*. Workspaces are similar to Knowledge Trees, but are more flexible and less strictly defined. They are useful when you only want to work with *part* of a Knowledge Tree, such as a topic set or taxonomy. For example, when you only want to create a topic set, and do not want to create a taxonomy, you might find a workspace more convenient than a Knowledge Tree because with a workspace you can create a topic set without creating a taxonomy.

The primary function of a workspace is to keep track of the files and settings used in a particular editing session. A workspace always has an associated locale and charset. Workspaces are stored in *topic workspace files*, with the filename extension `.tsw`.

You can use the *parts* of a workspace, such as a topic set, in other Verity applications. Table 2-1 shows the main differences between Knowledge Trees and workspaces.

Table 2-1 Comparison of Knowledge Trees and Workspaces

	Knowledge Trees	Workspaces
Taxonomies	Always contains one taxonomy.	May or may not contain a taxonomy.
	Intelligent Classifier automatically creates the taxonomy when it creates the Knowledge Tree. It calls it `workspace.tax` and stores it in the `IC` directory.	The taxonomy can be named anything (with a `.tax` extensions) and can be stored anywhere.
	You cannot close a taxonomy while the Knowledge Tree is in use.	The taxonomy can be closed and a new taxonomy can be opened.
Topic Sets	Always contains at least one topic set. The topic set might be empty, or it might contain topics that are stored searches for an individual collection. Any number of topic sets might be created for any single collection, but only one topic set can be active at a given time.	Can contain one or more topic sets.
	Intelligent Classifier automatically creates a topic set when it creates the Knowledge Tree. It calls it `topic` and stores it in the `IC` directory. The topic set is always stored in directory format.	The topic set(s) can be named anything and can be stored anywhere. The topic set(s) can be stored in either directory format or in `.otl` file format.
.tsw	Opened only by a Knowledge Tree	Can be opened directly from Intelligent Classifier.

Intelligent Classifier does not, by itself, make topics available to end users. It simply creates *topic sets* that you can use in other Verity applications. When you create more than one topic set, to use multiple topic sets in the same Verity application, you must also supply a knowledge base map (KBM) file.

A Knowledge Tree refers to, but does not modify, a collection. The attached collection, and the tools that maintain it, remain unaware of the Knowledge Tree. This uncoupled architecture lets you create and modify Knowledge Trees regardless of the construction of the underlying collections. You can create several Knowledge Trees for one collection; each Knowledge Tree provides a different organization for a single set of documents.

Understanding Parametric Indexes

Parametric indexes are indexes that allows retrieval of documents based on the values of parameters. It has the same purpose as a collection, which is to provide fast access to documents. As an example, the selection of red cars in California for model years 1995 and 1996 uses the car color and model year parameters. You can think of a parametric index as an in-memory extension to a collection that identifies documents matching the requested parametric selections. For more information, see "Parametric Index Creation" on page 183 and the *Verity K2 Parametric Developer Guide.*

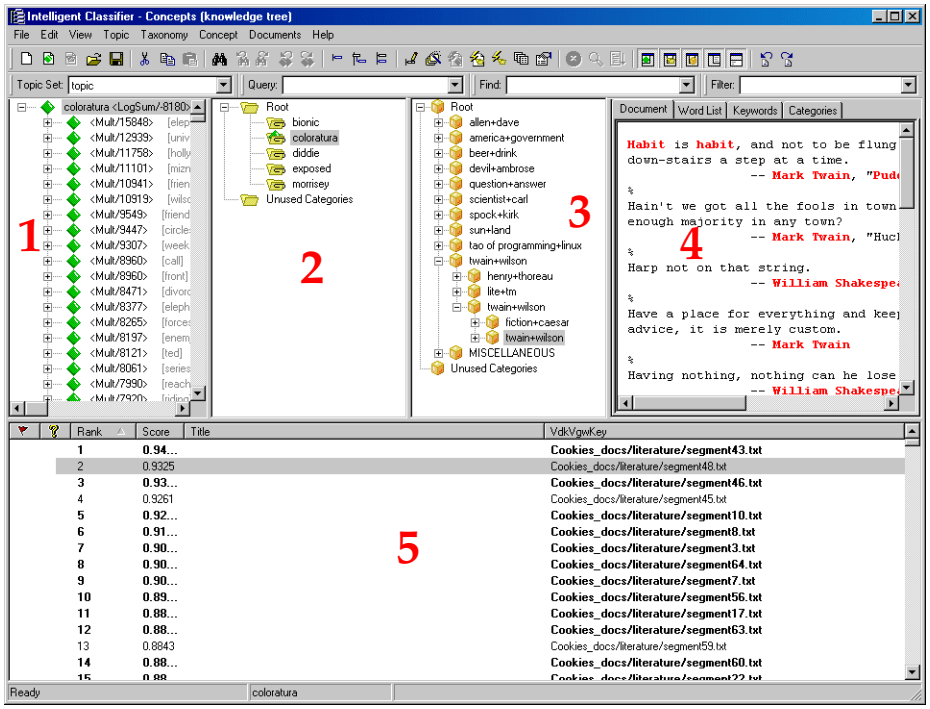
Intelligent Classifier User Interface

Intelligent Classifier is a tool for creating, viewing, editing, and testing Verity topics, taxonomies, Knowledge Trees, and parametric indexes. The Intelligent Classifier window contains a number of individual areas referred to as panes. Each pane represents a different view and a different functionality, so you can view and navigate with ease.

This section contains a brief introduction to the visual elements of the graphical interface, including descriptions for the following elements:

- Topic Pane
- Taxonomy Pane
- Concept Pane
- Document View Pane
- Search Results Pane
- Navigating Topic and Taxonomy Trees
- Additional User Interface Elements

Figure 2-1 Intelligent Classifier window



1. **Topic pane:** Displays topics.
2. **Taxonomy pane:** Displays categories in a taxonomy.
3. **Concept pane:** Displays a concept tree generated by concept mapping.
4. **Document View pane:** Displays the document selected in the Results pane or the return statistics of a search—a feature not discussed in this section.
5. **Search Results pane:** Displays the document results list. Double-click a result in the list to view it in the View pane.

The following sections describe the features and functionality of each pane in detail.

Topic Pane

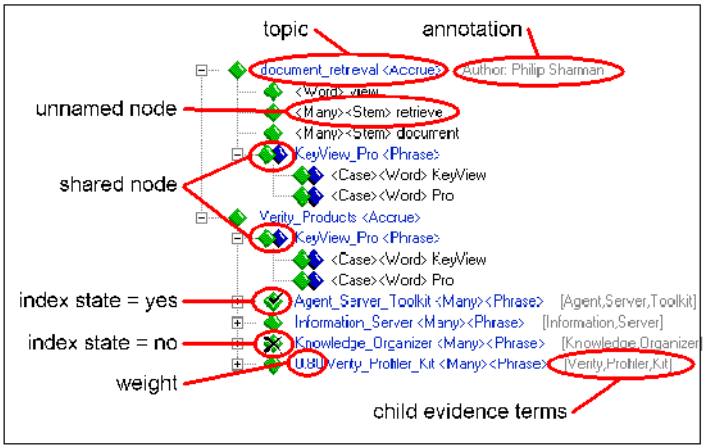
The Topic Pane displays the hierarchy of a topic set. This pane is described in more detail in "Understanding Workspaces" on page 32.

To toggle the display of the pane, choose **View | Topic Pane**, or click the toolbar button.

Elements in the Topic Pane

Intelligent Classifier uses color coding and symbols to represent different items in the topic tree.

Figure 2-2 Elements in Topic Pane



As shown in Figure 2-2, the elements in the topic pane are:

- Nodes—represented by green diamonds. Node names shown in blue are *topics*. Node names shown in black are unnamed nodes.
  - Green and blue diamonds together represent shared nodes.
  - Checkmarks on a node indicate that the index state is on. Crosses on nodes indicate that the index state is off.
- Annotations—shown to the right of nodes.
- Child evidence terms—shown to the right of collapsed nodes.
- Weights—weights assigned to items are shown in the node.

Taxonomy Pane

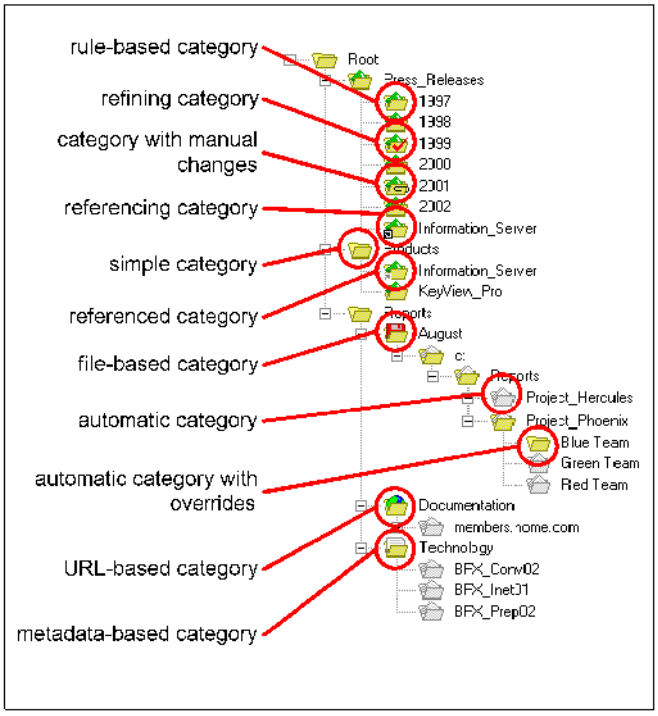
The Taxonomy Pane shows the hierarchy of the taxonomy. This pane is described in more detail in "Creating Categories" on page 69.

To toggle the display of the pane choose **View | Taxonomy Pane**, or click the toolbar button.

Elements in the Taxonomy Pane

Figure 2-3 shows the elements that make up the Taxonomy pane. A list of the color coding follows the figure.

Figure 2-3 Elements in Taxonomy Pane



- Folders containing green diamonds are associated with rules.
- Folders with checkmarks are refining categories.
- Folders with paperclip icons contain manually added, excluded, or ranked documents.
- Folders with large arrows in the corner are referencing categories. Folders with small arrows in the corner are referenced categories.
- Plain folders are simple categories.
- Folders containing floppy disks are the parents of file-based automatic categories.
- Folders containing globes are the parents of URL-based automatic categories.
- Folders containing documents the parents of are metadata-based categories.

- Folders shown in gray are automatically created categories.
- Yellow folders among gray ones are automatically created categories whose properties have been changed.

To edit category properties in a batch mode, select multiple categories, Right-Click and select **Properties**.

Concept Pane

Thematic mapping automatically extracts key concepts contained in a set of documents and organizes them into a hierarchy called a *concept tree*. The Concept Pane shows the hierarchy. This subject is described in more detail in "Thematic Mapping." Intelligent Classifier uses color coding and symbols to represent different items in the Concept Pane:

- Cubes are concept nodes
- Spheres are term nodes
- Cubes with green diamonds are concept nodes that are associated with rules.

Document View Pane

To preview a document, select it in the Search Results Pane. Intelligent Classifier displays the selected document and information about it in the Document View Pane.

The *Document* tab shows the selected document.

The *Word List* tab lists the non-trivial words in the document and the number of times each word appears. You can sort this list alphabetically or by frequency by right-clicking on the pane and using the shortcut menu.

This list contains all the words in the document except those contained in the `vdk30.stp` file. When you are using the `uni` locale, Intelligent Classifier uses the `.stp` file from *`IC_install`*`\common\uni` (where *`IC_install`* is the directory where Intelligent Classifier is installed). When you are using another locale, Intelligent Classifier uses the `.stp` file from that locale.

The *Keywords* tab uses a special algorithm to list all the important words in the document.

The shortcut menu for the Keyword tab provides a **Create Top-level Topic From Keywords** command. Using the keywords, this command automatically creates a topic designed to retrieve documents similar to the selected document.

The *Category* tab displays the list of categories that the selected document is currently assigned to along with the score for each. The list is sorted by score in descending order.

When viewing a document that you have flagged as relevant (or irrelevant) the words shown in the **Word List** tab and **Keywords** tab that are common to both the topic and the document are shown in green (or red).

After using the **Extract Keywords** command, the words shown in the Search Results Pane are color coded according to the *clusters* of related documents.

The **Documents** menu provides commands to let you sort the documents by their 'read' or 'new' state. It also lets you reset them all to their default states.

Sort the Search Results pane columns by clicking on their headers. A small arrow in the header shows which column is being used to sort the list, and whether it is sorted in ascending or descending order. To toggle the sort order, click again.

Resize and move the columns by dragging their headers.

Choose which columns to show in the Search Results pane.

Toggling the Pane On and Off

To toggle the display of the Document View Pane, select **View | Document View Pane**, or click the toolbar button.

Navigating from Highlight to Highlight

Intelligent Classifier shows each occurrence of a search term entered into the search bar in red. To display the next or previous occurrence, select **Documents | Previous Highlight** or **Next Highlight**, or click the toolbar buttons.

Note Lotus Notes documents with embedded OLE objects will not contain highlights.

Search Results Pane

The Search Results Pane shows the documents retrieved when you test a topic, run a query, or test a category. You can set the maximum number of such documents. You can also use the Filter Bar to restrict the documents that are shown. See "Filter List" on page 48.

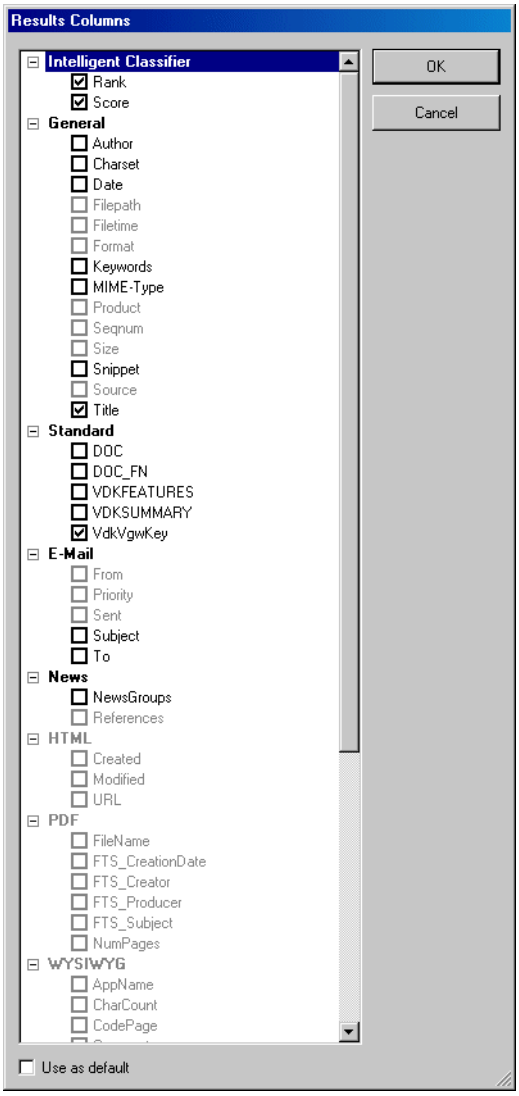
To toggle the display of the Search Results Pane, choose **View | Search Results Pane**, or click the toolbar button.

Selecting Columns to View

To choose which columns to display:

1. Choose **Documents | Customize Results Columns** or click the toolbar button. The Results Columns dialog appears.

Figure 2-4 Results Columns dialog



2. Check the items you would like shown in the Search Results Pane.

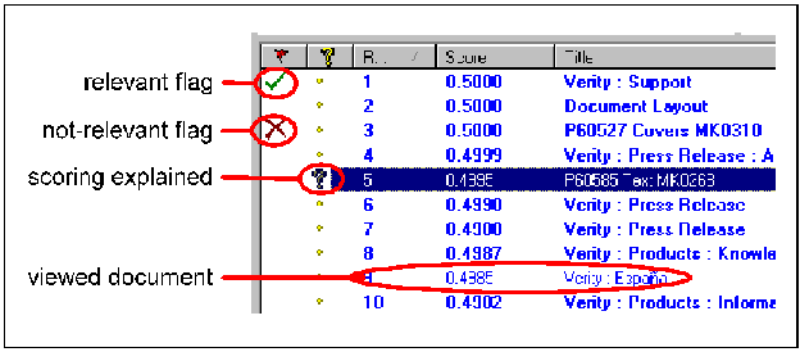
Note Fields that are not present in any of the currently open collections are shown in gray, but you can still select them.

3. Click **OK**.

Color Coding and Boldface

Intelligent Classifier uses color coding and symbols to represent different items in the Search Results Pane.

Figure 2-5 Color coding and boldface in the Search Results dialog



The following descriptions indicate how color and bold text are used in Figure 2-5.

- Green checkmarks show documents that have been flagged as *relevant* to the selected topic node or category node, depending on which is active. Red crosses mark ones that have been flagged as *irrelevant*. (These flags are for convenience; they do not affect the classification or search results in any way.)
- The question mark icon shows that the document's *scoring* is being explained.
- Documents shown in blue are ones that have been *retrieved for the first time*. Ones shown in black have been retrieved before.
- Documents shown in boldface are ones that have *not yet been viewed*. Ones shown in normal type have been viewed before.
- Asterisks beside a document's rank indicate that it has been manually ranked or manually categorized.

This information is saved with the workspace so it is still available when you close the workspace and re-open it later.

To reset this information, choose the appropriate command from **Documents | Document States**.

When you choose **Reset All Documents To Default**, this also resets the relevance flags, described later.

Sorting Columns

To sort a column, click the column's title box. Click again to toggle between ascending and descending order.

Resizing and Rearranging Columns

In Intelligent Classifier you can move and resize the fields' columns.

To resize a column, move the cursor to the column's title boundary until the cursor changes to a double-headed arrow, then drag the column's boundary to the right or left.

To move a column, click and drag the column's title to the right or left until it is in the correct position.

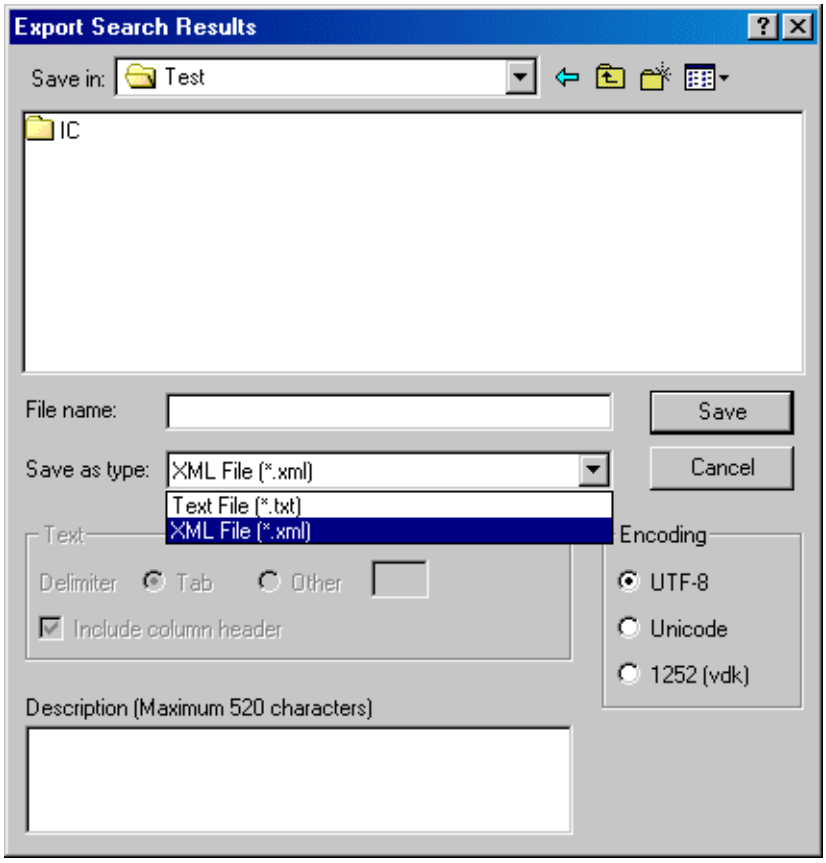
Exporting Search Results

You can export the search results displayed in the results pane to an XML file or a plain text table format file.

The following procedure demonstrates how to export the search results from a test.

1. Select **Documents | Export Search Results**. The Export Search Results dialog box appears.

Figure 2-6 Export Search Results dialog



2. Type a name for the file in the **File name** edit box.
3. Enter either Text or XML as the format you wish to save to in the **Save as type** drop-down list box.
4. Select one of three encoding schemes for the results file.

   This can be either UTF-8, Unicode, or the default code page of the current VDK session. The actual code page number depends on the VDK session locale.
5. Click **Save** to export the results.

XML Format

Selecting XML as the file save type exports all entries in the results window to an XML file. Each document's entry contains all columns selected in the results pane, except *VdkVgwKey*. Each entry appears in a "Document" element and all of its fields as

sub-elements of "Document". *VdkVgwKey* is an attribute of the *Document* element. You can include an optional *Description* element. The following example shows some search results exported in XML format.

```
<?xml version="1.0" encoding="ISO-10646-UCS-2"?>
<VICSearchResult>
    <Description>
        unicode: &lt;&quot;&gt;&apos;&amp;
    </Description>
    <Document VdkVgwKey="Cookies_docs/science/segment157.txt">
        <Rank>1</Rank>
        <Score>    0.4255</Score>
        <Title></Title>
    </Document>
    <Document VdkVgwKey="Cookies_docs/science/segment21.txt">
        <Rank>2</Rank>
        <Score>    0.4175</Score>
        <Title></Title>
    </Document>
    <Document VdkVgwKey="Cookies_docs/science/segment113.txt">
        <Rank>3</Rank>
        <Score>    0.4175</Score>
        <Title></Title>
    </Document>
</VICSearchResult>
```

Plain Text Format

Saving the search results as text exports the results into a plain text delimited table format file. Plain text output enables you to import your results into applications such as MS Word or Excel.

You must select a column delimiter that can be either a tab or a character of your choosing.

To include column headers (field names), check the **Include column header** check box.

The following shows some search results exported in plain text format. The default tab delimiter was used.

```
RankScoreTitleVdkVgwKey
1       0.4255Cookies_docs/science/segment157.txt
2       0.4175Cookies_docs/science/segment21.txt
3       0.4175Cookies_docs/science/segment113.txt
4       0.4173Cookies_docs/science/segment24.txt
5       0.4173Cookies_docs/science/segment116.txt
6       0.4173Cookies_docs/cookie/segment52.txt
```

```
7       0.4173Cookies_docs/cookie/segment229.txt
8       0.4084Cookies_docs/science/segment42.txt
9       0.4084Cookies_docs/science/segment161.txt
10      0.4084Cookies_docs/definitions/segment90.txt
11      0.4084Cookies_docs/definitions/segment231.txt
12      0.4081Cookies_docs/science/segment85.txt
13      0.4081Cookies_docs/science/segment154.txt
```

Using Relevance Flags

You can use the left-most column, headed by the flag icon  , to set a green checkmark or red cross beside each document. These flags help you to identify at a glance which documents are relevant or irrelevant to topic nodes or taxonomy nodes.

Intelligent Classifier can use one set of flags for all topics in the topic set, or use a different set for each topic. When you change the setting for this property, Intelligent Classifier remembers the relevance flags that were set for the original property and restores those flags when you change the setting back again. You can keep one set of flags for all topics (using the "All topics use the same relevance settings" property) and a different set of flags for each topic (using the "Each topic has its own relevance setting" property).

To set or clear the relevance flag for a document, click in the relevance column beside the document.

Clicking again cycles the icon between the checkmark, the cross, and the blank state.

To clear the relevance flags for all the documents, choose the appropriate command from **Documents | Document States**.

When you have "Each topic has its own relevance settings" set in the properties then resetting the relevance flags only affects the flags for the current topic. This includes the **Reset All Documents To Default** command.

When you choose **Reset All Documents To Default**, the color coding and boldfacing are reset.

To show all the documents that have been flagged as relevant or irrelevant for a topic or a category, select the topic or category, then choose **Documents | Return Flagged Documents**.

When you have **All topics use the same relevance settings** set in the properties, Intelligent Classifier retrieves all flagged documents, whether or not they match the last query that was run. This lets you look at all the documents you've flagged without having to remember the topics or queries you used to find them originally. (They show their score from the last search, or 1 when they were not retrieved by the last search.)

When you have **Each topic has its own relevance settings** set in the properties, then Intelligent Classifier retrieves all flagged documents for the last query that was run.

Note Each topic set in a workspace has its own set of relevance flags. So, when you change the active topic set, Intelligent Classifier shows the flags that have been set for the new topic set.

Multiple documents in the result pane can be marked as relevant, irrelevant, or cleared with their flags in a batch mode. To do this, select multiple documents in the Result List Pane, right-click, and select the desired action.

Navigating Topic and Taxonomy Trees

You can navigate up and down the topic or taxonomy tree by using the Up Arrow and Down Arrow keys or the mouse.

Expanding and Collapsing Nodes and Categories

Intelligent Classifier provides three ways to expand and collapse nodes and categories:

1. Click on the plus signs or minus signs in the tree.

   This expands and contracts one level of the tree below the selected node/category, just as in Windows Explorer.

2. Select a node/category, and choose either **Edit | Expand** or **Edit | Collapse**, or click one of the toolbar buttons.

   This expands or contracts the selected node/category as much as possible.

3. Select the node/category, and press the Left Arrow or Right Arrow keys.

The first method provides you with more control over which parts of the tree are expanded. Intelligent Classifier also remembers which nodes/categories have been expanded this way and maintains that setting as you collapse and re-expand their parents.

Note The more items that are expanded, the more resources Intelligent Classifier requires. When you are using very large trees, it is recommended that you only expand the items you need.

Additional User Interface Elements

The following sections describe additional user interface elements in Intelligent Classifier.

Topic Set List

The Topic Set list provides a drop-down list of all topic sets in the workspace.

Note Whether Intelligent Classifier shows the Topic Set list can be controlled though the Toolbars settings.

Find List

To list all the topics in the topic set, click the Find list, located beneath the toolbar.

When you type a letter, the list scrolls to topic names beginning with that letter. Typing the same letter again moves down the list. When you select a name from the list, Intelligent Classifier selects and moves to that topic in the topic tree. (When it is a shared topic, you can find other occurrences by using the **Previous Use of Node** or **Next Use of Node** commands.)

You can also find topic names, and other items, through the Find command.

You control whether Intelligent Classifier shows the Find list from the Toolbars settings.

Query List

The Query list enables you to see what users' queries entered in a Verity application, such as Verity K2 Enterprise, will retrieve. K2 processes the queries using Verity Query Language (VQL).

To use the **Query** Bar:

1. Type the query into the **Query** Bar.
2. Either:
   - Press **Enter**.
   - Choose **Topic | Run Query**.
   - Type **Ctrl+R**.
   - Click the **Toolbar** button.

The Search Results pane shows the documents that are retrieved by that query.

For example, when the current query parser is the simple parser, then typing "a AND b" into the Query list finds documents that contain "a" and "b".

The Query drop-down list shows your previous queries and lists, in braces ("{", "}"), all the topics that have been tested. You can re-run a search by selecting it from the Query drop-down list and pressing **Enter**.

You can perform a *null search* (retrieve all documents in the collection) by deleting anything that is in the Query Bar and pressing **Enter**.

The VQL syntax that Intelligent Classifier accepts is determined by the current query parser.

You can also use VQL syntax to create topics and query-based topics. For more information on VQL, see the*Verity Query Language and Topic Guide*.

Filter List

In the Filter list you can restrict the documents to be searched to a subset of the attached collection(s).

To use the **Filter** Bar:

1. Enter the filter, using Verity Query Language, in the **Filter** Bar. Examples of filters that can be entered in the **Filter** Bar (using the Boolean Plus query parser) are:

   ```
   Author <CONTAINS> Smith

   Title <STARTS> Verity

   Date >= 28/May/1934
   ```

2. Either:
   - Test a topic
   - Enter a query in the **Query** bar
   - Test a category

The Results Pane shows only documents that match the search *and* the specifications of the Filter Bar. (This only affects the view in the Search Results Pane. It does not affect what end users see when they use the query, topic, or category.)

The syntax that Intelligent Classifier accepts is determined by the current query parser.

To stop using the filter, erase (by backspacing over it) the contents of the Filter list. You can control whether Intelligent Classifier shows the Filter list from the Toolbars settings.

Status Bar

The **status** bar is located at the bottom of the window. The **Status** Bar shows feedback from Intelligent Classifier. The left section explains what Intelligent Classifier is currently doing. The center section gives progress information when Intelligent Classifier is performing a lengthy task (such as executing a search). The right section gives feedback about why Intelligent Classifier has done certain actions. When Intelligent Classifier beeps while you are using assist mode, look here to see what error it has detected.

Shortcut Menus

To view Intelligent Classifier's shortcut menus, right-click in the Topic Pane, Taxonomy Pane, Document View Pane, toolbar, or on column headers in the Search Results Pane. The shortcut menus allow quick access to the most common commands in that pane.

Working With Knowledge Trees and Workspaces

The top of the Intelligent Classifier window shows whether you are working with a Knowledge Tree or a workspace.

- When you are working with a Knowledge Tree, Intelligent Classifier shows the name of the Knowledge Tree followed by `[knowledge tree]`.
- When you are working with a workspace, Intelligent Classifier shows the name of the workspace file, the name of the active topic set, and the name of the taxonomy file.

Defining a New Knowledge Tree

To define a new Knowledge Tree, use Intelligent Classifier's Classification Wizard.

1. The wizard starts automatically when you launch Intelligent Classifier unless the startup options have been changed. If the wizard is not already started, from the Intelligent Classifier window, choose **File | Classification Wizard**.
2. In the Welcome dialog, click **Next**.

   A dialog asks you to name the Knowledge Tree and provide a location.
3. Type a name and specify a location for the Knowledge Tree.

To specify the location, either:

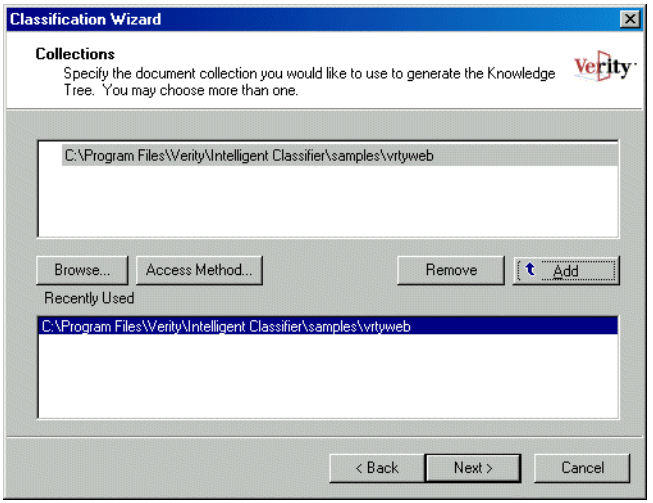
- Type the path directly in the **Location** field.
- Or, use the browse button ("...") beside the Location field to navigate to the desired folder.

Intelligent Classifier automatically creates a topic set and taxonomy in the Knowledge Tree's `IC` directory. The topic set is named `topic` and the taxonomy is named `workspace.tax`.

4. Click **Next**.

   A dialog containing a list of existing collections appears.

Figure 2-7 Classification Wizard



5. Select the Verity collection(s) of documents that the Knowledge Tree will search.

   Collections can be added and removed at any time from Intelligent Classifier, not just when creating a new Knowledge Tree.

Note Removing a collection from Intelligent Classifier does not delete the collection from the drive it resides on.

You can change the way Intelligent Classifier accesses the documents in the collection by clicking Properties.

6. Click **Next**.

An Automatic Classification dialog appears. Use this dialog to select a method to automatically classify the document or select **None** to bypass automatic classification.

Clusters—You can cluster the documents into groups of similar documents.

Metadata—Generate new categories based on the metadata (fields) of the documents in the collection. For example, you can create a set of new categories where each category retrieves documents by a certain author.

File Path or URL—Generate new categories based on path information (either the file path or the URL) of documents in the collection. This creates a hierarchy of categories that mirrors the path hierarchy.

7. Click **Next**.

   You will see an additional dialog whose contents will vary depending on the categorization method you chose, unless you chose **None**. Use this dialog to provide information about the automatic classification scheme that you selected.

8. Click **Next**.

   A dialog indicates that you are completing the automatic classification process.

9. Click **Finish**.

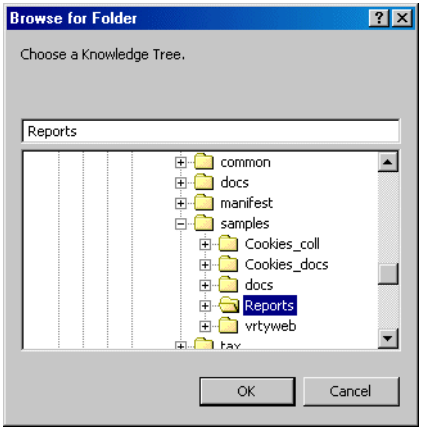
   Intelligent Classifier creates the new Knowledge Tree, topic set, and taxonomy.

Opening and Generating a Knowledge Tree

To open a Knowledge Tree:

1. Choose either **File | Open Knowledge Tree**, press **Ctrl+O**, or click the **Open** button.

2. Select the “Reports” Knowledge Tree, created in the previous section.



To generate a Knowledge Tree:

1. Choose **File | Generate Knowledge Tree**.

   This generates the files and directories used by other Verity applications when they work with Knowledge Trees. Generating a Knowledge Tree is separate from the task of creating the Knowledge Tree, described in the previous section.

   This command has the same effects as if you used `ktmgr` and specified the appropriate files inside the `IC` directory.

Note To speed the opening of a knowledge tree, uncheck the **Save document retrieval status** option under **View | Options**. See "Setting Intelligent Classifier Options" for more information.

Note Unlike topic sets, Knowledge Trees contain references to specific collections. The information in the Knowledge Tree records which documents in those collections will appear in each category. So, before you choose Generate Knowledge Tree, ensure you have the same collections attached that end users will be using when they use this Knowledge Tree.

Saving and Closing a Knowledge Tree

To save the Knowledge Tree:

1. Choose either **File | Save All**, press **Ctrl+S**, or click the **Save** button.

To close a Knowledge Tree:

1. Choose **File | Close Knowledge Tree**.

Moving a Knowledge Tree

When you need to move a Knowledge Tree to a new location, Verity recommends that you use the Copy command or use one of the command-line tools to relocate the Knowledge Tree. This preserves database integrity in a case where indexing operations are in progress. When the location of the primary collection or external taxonomy database is changed in the course of the move, update the Knowledge Tree's `dir.cfg` configuration file to reflect the new location(s).

Knowledge Tree databases store data in a platform-independent manner. This allows a Knowledge Tree created on one operating system platform to be copied to another operating system platform.

Using Other Topic Sets

A Knowledge Tree can contain only one taxonomy, but it can use more than one topic set. There are several ways to use additional topic sets in Intelligent Classifier.

- You can import a new topic set.

  This merges it with the active topic set.

- You can open the topic set.

  This makes the topic set available and stores a link to the topic set in the `.kbm` file in the `IC` directory but does not copy the topic set to the `IC` directory. This lets more than one Knowledge Tree share the same topic set. When you move the Knowledge Tree, the link will still refer to the topic set as long as the topic set can be reached from new location.

  It is possible that the link will not work in the new location. For example, when one person uses a topic set on a network drive and the Knowledge Tree is moved to another

location, then Intelligent Classifier will not be able to open the topic set if that drive is not visible from the new location. In this case, administrators can remove the topic set and re-add it.

- You can move the topic set into the `IC` directory.

1. Move or copy the topic set's directory or `.otl` file into the `IC` directory.
2. Use Intelligent Classifier's Open Topic Set command to open the topic set.

Remote Administration

When you create a Knowledge Tree using Intelligent Classifier's Classification Wizard, Intelligent Classifier creates a directory for the Knowledge Tree. This directory has the same name that was specified as the name of the Knowledge Tree. Inside that directory, it also creates a directory called `IC`. This directory holds files that are used by Intelligent Classifier.

The fact that all the relevant Intelligent Classifier files are contained inside the Knowledge Tree enables multiple administrators to use Intelligent Classifier to edit the same Knowledge Tree.

Note When you are using collections mapped to different networked drives, you might need to remove and replace the collections so that Intelligent Classifier can locate them.

Intelligent Classifier's "Generate Knowledge Tree" command builds the other files and directories that are used by Verity K2 Enterprise. Once the files and directories are built, the Knowledge Tree is ready to use in K2.

`ktmgr` does not create the `IC` directory. This means that Intelligent Classifier cannot open a Knowledge Tree that was created by `ktmgr`. It can, however, open the taxonomy file and related files that were used by `ktmgr`.

About Lock Files

While Intelligent Classifier is editing a Knowledge Tree, it locks it so that other users cannot edit it at the same time. It does this by creating a `lock.ini` file in the Knowledge Tree's `IC` directory. Normally, Intelligent Classifier removes the lock when you close the Knowledge Tree. When you need to manually remove the lock, you can do so by deleting the `lock.ini` file.

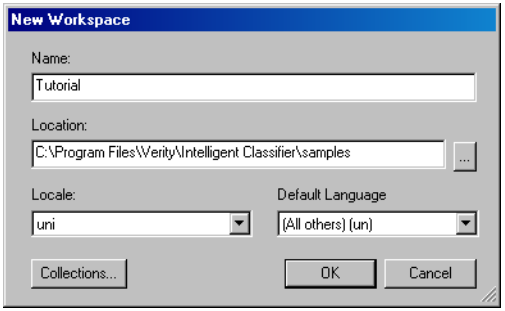
Creating a New Workspace

To create a new workspace:

1. Choose either **File | New Workspace**, type **Ctrl+N**, or click the toolbar button.

The **New Workspace** dialog appears.

Figure 2-8 New Workspace dialog



2. Enter a name and location for the workspace.

   To specify the location, either:

   - Type the path directly in the **Location** field.
   - Use the browse button ("...") beside the Location field to navigate to the desired folder.

   By default the **Locale** is `uni`. The **Default Language** drop-down list is valid only if the **Locale** is `uni`, and is used for language-specific stemming and tokenization when the sub-language is not specified.

   Note If you are licensed for and have installed the `englishx` locale, `englishx` is the default locale.

   Intelligent Classifier creates a `.tsw` file with that name in the location you chose.

3. To add a collection at this point, click **Collections**. You can also add collections later.
4. Click **OK**.

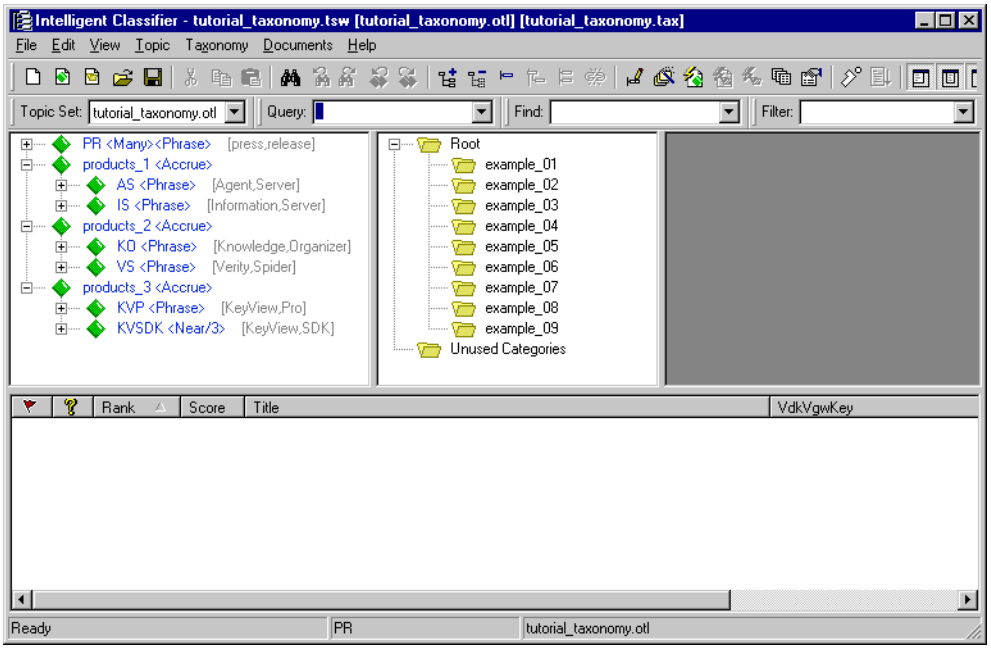
The workspace is created.

You can now add topic sets and a taxonomy.

Opening a Workspace

To open an existing workspace, do the following:

1. Select **File | Open Workspace**.
2. Open `tutorial_taxonomy.tsw` located in *`IC_install`*`\samples\` (where *`IC_install`* is the directory where Intelligent Classifier is installed, typically `C:\Program Files\verity\intelligent classifier`).



3. When necessary, choose **View | Taxonomy Pane** to display the Taxonomy Pane.
4. Expand the topics `products_1`, `products_2`, and `products_3` as shown.
5. Select **File | Add/Remove Collections** to add the `vrtyweb` collection.
6. When the Collections window opens click **Browse** and select *`IC_install`*`\samples\vrtyweb`.

Where *IC_install* is the Intelligent Classifier installation directory (typically `C:\Program Files\verity\Intelligent Classifier`).

Saving a Workspace

To save the workspace, choose either **File | Save All**, type **Ctrl+S**, or click the **Save All** button.

Closing a Workspace

To close a workspace, choose **File | Close Workspace**.

Renaming Workspaces

To save the workspace under a different name, choose **File | Save Workspace As**.

When you do this, Intelligent Classifier creates a new workspace (`.tsw`) file with the name that you supply in the dialog and uses that for the current workspace.

Adding a Collection

The following steps load a collection.

1. Select **File | Add/Remove Collections**. The Collections dialog appears.

Note When you have recently-used collections, those collections appear in the space "Recently Used" in the Collections dialog. If you have no recently-used collections, there will be none available in this dialog.

2. In the Collections, dialog, click on the **Browse** button to display a file tree that you can browse to select a collection.
3. Select *IC_install*`\samples\vrtyweb` (where *IC_install* is the Intelligent Classifier installation directory, for example `C:\verity\Intelligent Classifier`).
4. Select the collection and click **OK**.
5. Repeat steps 2 to 4 to add more collections, if additional collections are available.

6. Click **OK** to close the Collections dialog.
7. Authenticate secure collections.

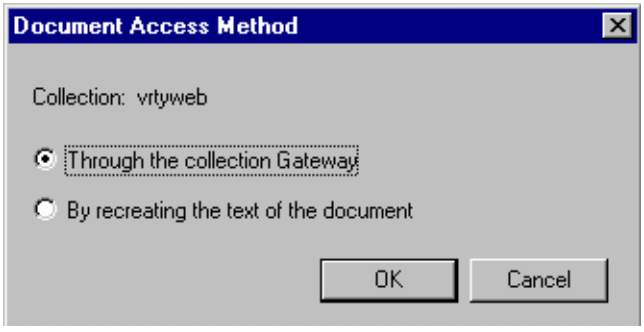
Note You can only add collections that use the same locale as the original attached collection. You are not allowed to mix locales within the same workspace.

Setting the Document Access Method

The document access method tells the Intelligent Classifier how to retrieve document contents for view.

Select the **Access Method** button to choose how the documents will be retrieved when they are displayed in the **Document View** pane.

Figure 2-9 Document Access Method dialog



- If the repository on which the collection is built is available, select **Through the collection Gateway**.
- If the repository on which the collection is built is not available, then select **By recreating the text of the document**.

  In this case, Intelligent Classifier does not access a document directly but *recreates* it from the word list in the collection. The document is displayed as unformatted text because the collection does not store any formatting information.

Perform steps 1 through 3 on the first collection, repeat them on a second collection (if a second collection is available) and proceed to the end of the steps.

1. Because the documents are stored locally, select **Through the collection Gateway**, and click **OK**.

2. Click **OK** to close the Collections Properties window.
3. If the collection is a *secure collection* an authentication dialog appears.

   a. Enter the required information to access the secure collection.

   b. Click **OK** to close the authentication dialog.
4. Click **OK** to close the Collections window.

Creating a New Topic Set

1. Choose **Topic | New Topic Set**.

   The New Topic Set dialog appears.
2. In the Name field, enter `tutorial` for the name of the topic set.

Note The Name field should be populated with the name of the workspace, when it is not name the topic set as above.

3. In the **Location** field, enter a suitable location.

   The recommended location is the default location for the `.tsw` file, which is: *`IC_install`*`\samples` (where *`IC_install`* is the Intelligent Classifier installation directory, for example `C:\verity\Intelligent Classifier`). However, if you choose not to accept the default, ensure you make a note of where saved the topic set, you will need it later.
4. Select **Topic | Save Current Topic Set.**
5. Select **Topic | Close Current Topic Set** to close the New Topic Set window.

You can also launch Intelligent Classifier by clicking on a `.tsw` file or `.otl` file in Windows Explorer. (Clicking an `.otl` file also automatically creates a `.tsw` file with the same name.)

Removing A Collection

To remove a collection:

1. Choose **File | Add/Remove Collections**, or click the **Add/Remove** icon. The Collections dialog box opens.

2. Select the collection in the list on the Collections dialog box and click **Remove**.

You have now completed the tutorial. You might want to create categories from your topics. For information on how to do this, see Part III, "Populating Taxonomies."

Setting Intelligent Classifier Options

Intelligent Classifier's options apply to all Knowledge Trees or workspaces. (You can also set properties for each individual Knowledge Tree or workspace.)

To set Intelligent Classifier's options, choose **View | Options**, or click the **Options** icon.

The Options dialog box appears.

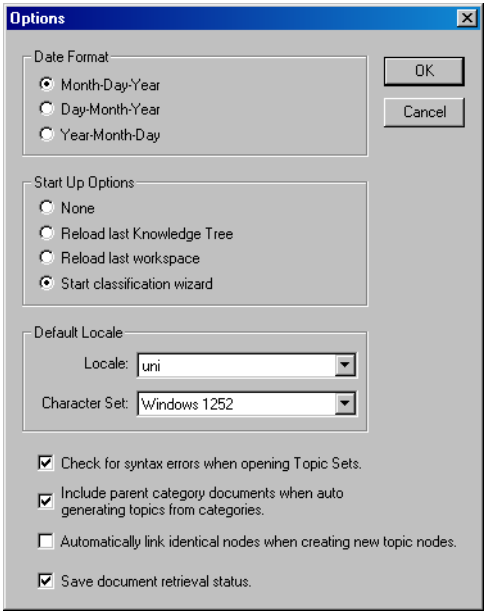


Table 2-2 Options Dialog Box User Interface Elements

Option	Effect
Date Format	Sets the format that Intelligent Classifier uses to parse dates (such as those that can be entered with the `<Field>` operator) if they are ambiguous.
Start Up Options	Determines the screen you see when you first start Intelligent Classifier. ■ If you select *None*, Intelligent Classifier opens with no files loaded. ■ If you select *Reload last Knowledge Tree*, Intelligent Classifier automatically opens your most recently used Knowledge Tree. ■ If you select *Reload last workspace*, Intelligent Classifier automatically opens your most recently used workspace. ■ If you select *Start classification wizard*, Intelligent Classifier starts the Classification Wizard with which you can create a new Knowledge Tree and automatically generate a new topic set and taxonomy.
Check for syntax errors when opening Topic Sets.	If this option is selected, Intelligent Classifier ignores problems with syntactically invalid OTL files when you open a topic set. NOTE: To do this, Intelligent Classifier might need to delete parts of the topic set, so this option should be used with caution. Turning this option off is the same as turning off precedence checking in `mktopics`.
Default Locale	Determines the locale and character set used when you create a new Knowledge Tree or workspace. After you create a workspace, you cannot change its locale. The current workspace's locale is displayed in Intelligent Classifier's General properties. A *character set* is used for viewing documents; however, Intelligent Classifier's viewing component regards it as a suggestion and might not use it if the viewing component determines that a document is in another character set.
Save document retrieval status	If this checkbox is checked, the search/view status of all documents in collections loaded in the current workspace will be saved when the workspace is closed and will be restored when the workspace is opened again. However, this will increase the workspace file size and take more time to save/load the workspace. If you do not need document retrieval status information to persist across sessions, unchecking this option can improve workspace save/load efficiency.

Using Locales

Intelligent Classifier supports Verity *locale*s, including multibyte locales. Locales provide support for collections containing foreign language documents.

In **File | Properties | General**, the **Locale** and **Default Language** drop-down lists are read-only. They display the locale and language for the current workspace. **Charset** can be set, and affects document viewing in the current workspace. **Charset** is effective when clicking a document for viewing; however, you will not see the view change in WYSWYG documents with embedded encoding information (always used). Text/Plain documents are handled by Internet Explorer, where you can right-click in the viewing pane to change **charset**.

If you want to change the default locale and character set, use Intelligent Classifier's options (**View | Options**). In **View | Options**, both **Locale** and **Charset** can be set; however, the new setting is used in creating new workspaces or knowledge trees, and is not effective until you open a new workspace. As in **File | Properties | General**, **Charset** is only used for viewing. When attaching a collection to Intelligent Classifier, the collection locale *must* match the Intelligent Classifier workspace locale or an error message is displayed.

3

Installing Intelligent Classifier

The following section describes the installation procedure for Verity Intelligent Classifier.

Verity Intelligent Classifier runs only on the Microsoft Windows NT V4.0 and Microsoft Windows 2000 and XP platforms.

Windows Intelligent Classifier Installation Procedure

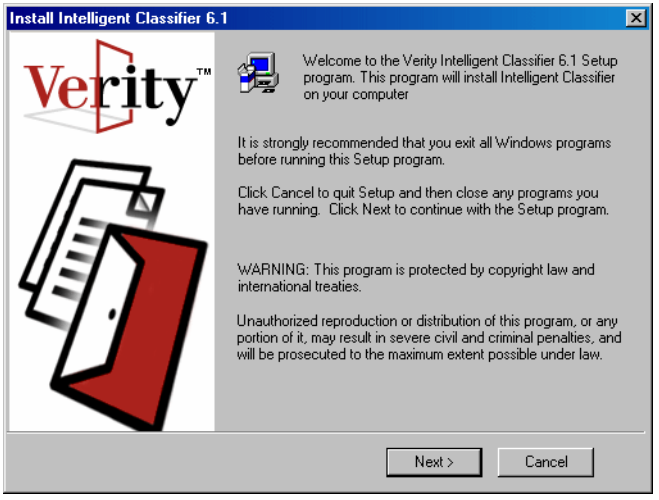
This section describes some easy steps to follow to install the Verity Intelligent Classifier software.

You need a license key for Verity Intelligent Classifier.

Note Installing Intelligent Classifier on the same machine as another Verity product, including older versions of Intelligent Classifier, may cause conflicts.

Starting the Intelligent Classifier Installer

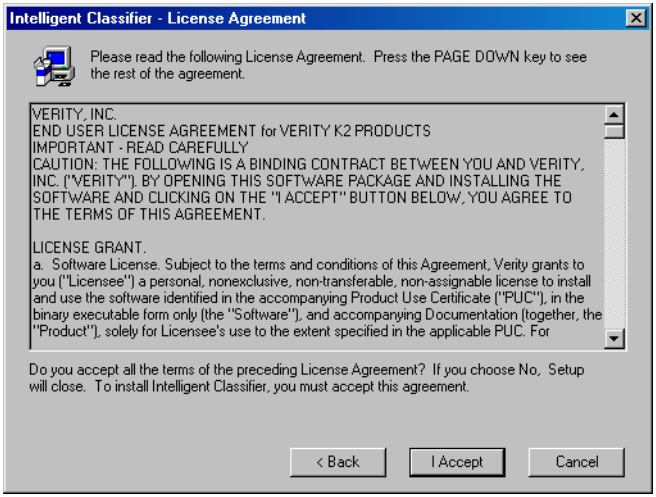
1. Insert the Verity K2 DVD and navigate to the `IntelligentClassifier` directory.
2. Double-click on **Setup** to run the setup program.
3. The Intelligent Classifier Welcome dialog appears.



Quit any other programs you are running before proceeding with this install.

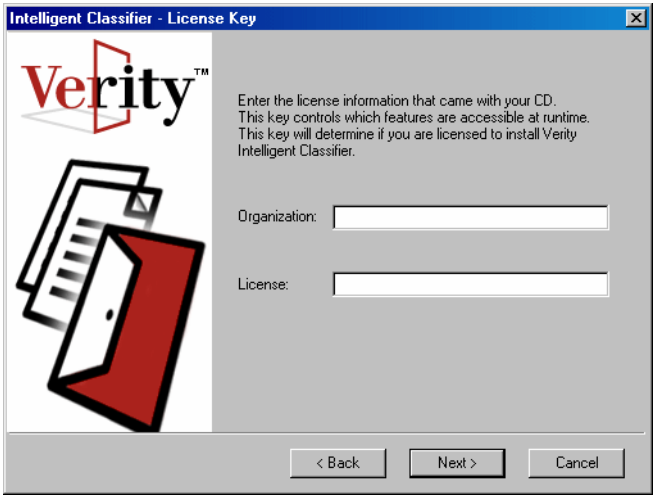
4. Click **Next**.

   The Verity Software License Agreement dialog appears.



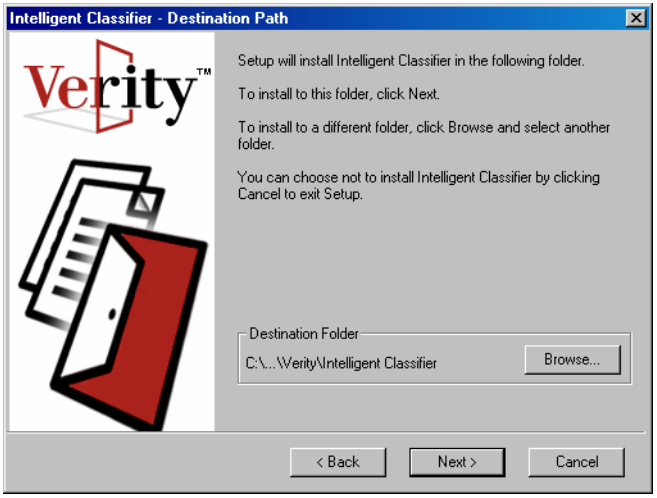
5. Read the license agreement and click **I Accept** to proceed.

   The Intelligent Classifier - License Key dialog appears.



6. Enter the name of your **Organization**.
7. Enter the license key that you were given and click **Next**.

   The Destination Path dialog appears.



From this dialog, you determine the directory path where Intelligent Classifier will be installed. You can choose to install Intelligent Classifier in a directory other than the default by clicking **Browse** and selecting a directory. The default directory is:

```
Program Files\Verity\Intelligent Classifier\
```

8. Accept the default install directory or type a new directory and click **Next** to begin copying files.

   A progress bar appears.

   Once the installation process has finished, the **Setup Complete** dialog appears.

9. To complete the installation click **Finish**.

PART I

Building Taxonomies

This part describes the features of taxonomies, including:

- Creating Categories
- Thematic Mapping

4

Creating Categories

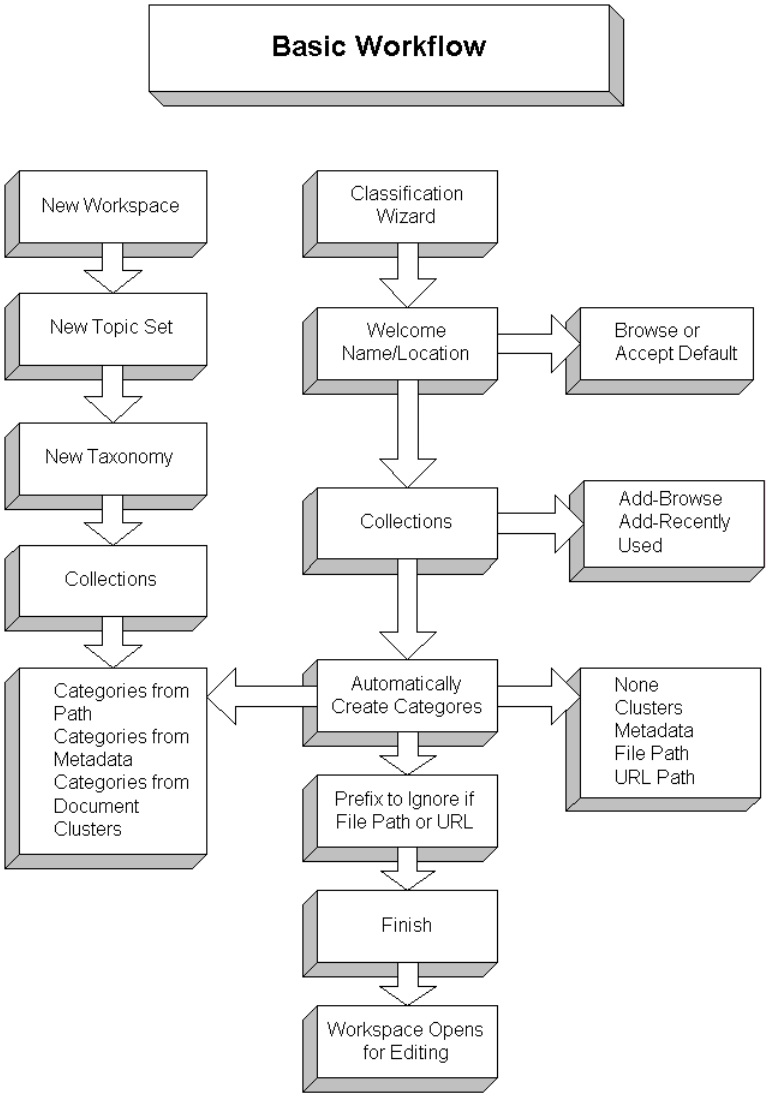
This chapter gives a hands-on overview of Intelligent Classifier's main taxonomy and category-editing features. To view a table of terms that you encounter during this tutorial, refer to "Understanding Workspaces."

This tutorial contains the following sections:

Basic Workflow

The following Basic Workflow Diagram shows the basic workflow for creating and searching Knowledge Trees.

Figure 4-1 Basic Workflow Diagram



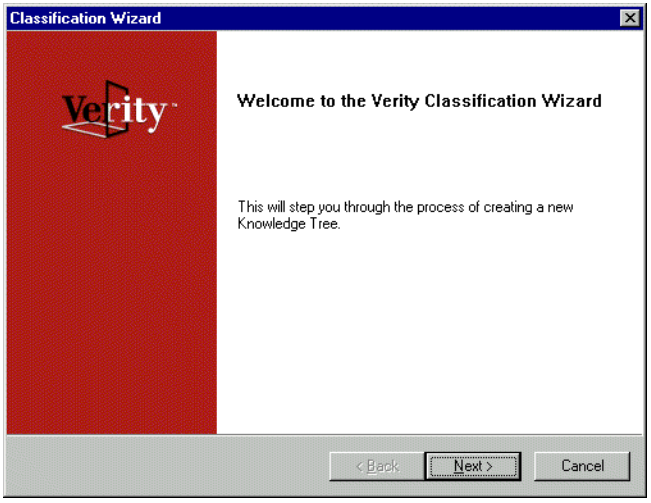
This section discusses how to create, open, close, and save Knowledge Trees using the Classification Wizard.

Defining a Knowledge Trees Using the Wizard

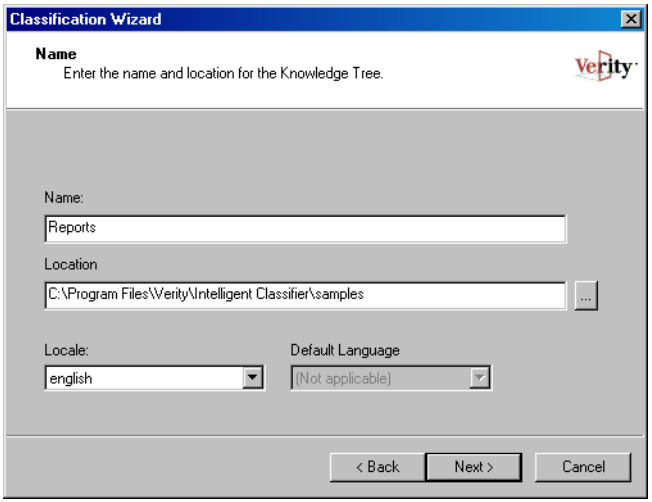
This tutorial takes you through a series of steps you need to complete to create a new Knowledge Tree, using the Verity Intelligent Classifier *Classification Wizard*.

1. The wizard starts automatically when you launch Intelligent Classifier (unless the startup options were modified). If the wizard is not already started, choose **File | Classification Wizard** in the Intelligent Classifier window.

Figure 4-2 Classification Wizard welcome dialog



2. In the Classification Wizard welcome dialog (Figure 4-2), click **Next**.
3. In the **Name** field type a name for the Knowledge Tree, for example, reports.



4. In the **Location** field specify a location for the Knowledge Tree.

   To specify the location, either type the path directly in the Location field or use the browse button (. . .) to the right of the Location field to navigate to the desired folder

   For this exercise ensure the path is:

   install_dir\Verity\Intelligent Classifier\samples

   Where `install_dir` is the directory where Intelligent Classifier is installed.

   Intelligent Classifier automatically creates an empty topic set and taxonomy in the Knowledge Tree's `IC` directory. The topic set is named `topic` and the taxonomy is named `workspace.tax`.

5. In the **Locale** field, select `english` for this exercise. The **Locale** choices are:

   - uni
   - english

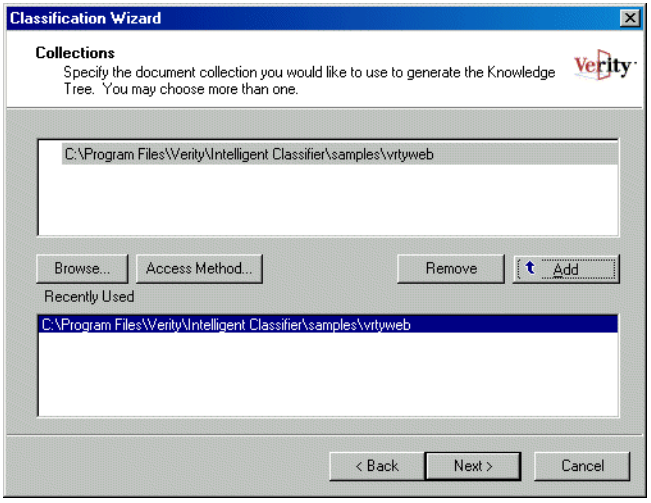
   If you select `uni`, the **Default Language** drop-down list contains a list of the languages available. An example of the active **Default Language** list is:

6. Click **Next**.

Creating Categories

In the next few dialogs, you create categories from collections of documents. The Collections dialog appears.

1. In the lower part of the dialog, click to select the desired Verity collection and then click **Add** to move the Verity collection to the list of documents that the Knowledge Tree searches.

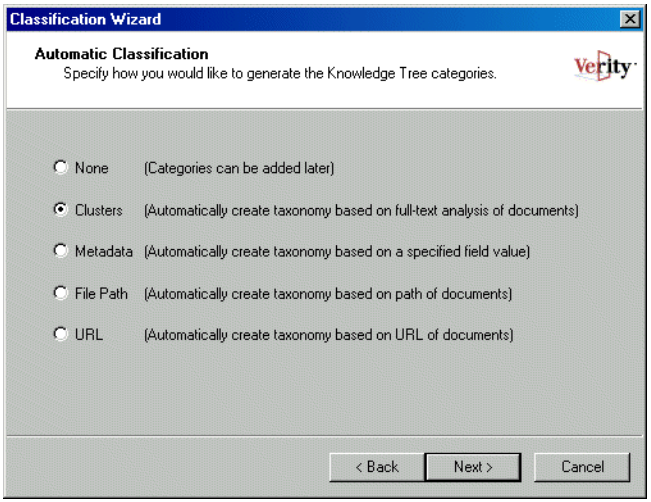


The location of the Verity collection is:

install_dir\Verity\Intelligent Classifier\samples\vrtyweb
Where *install_dir* is the path where Intelligent Classifier is installed.

Collections can be added to and removed from the workspace at any time, not just when creating a new Knowledge Tree.

2. Click **Next**.

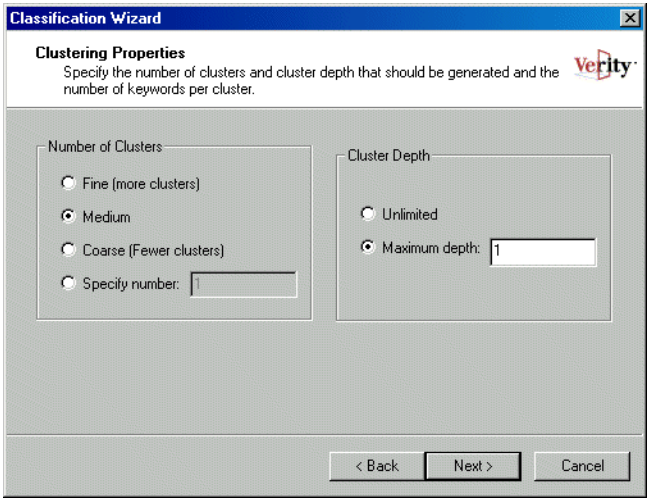


To automatically create a taxonomy based on the collection, select the method that Intelligent Classifier uses. The following list describes the choices.

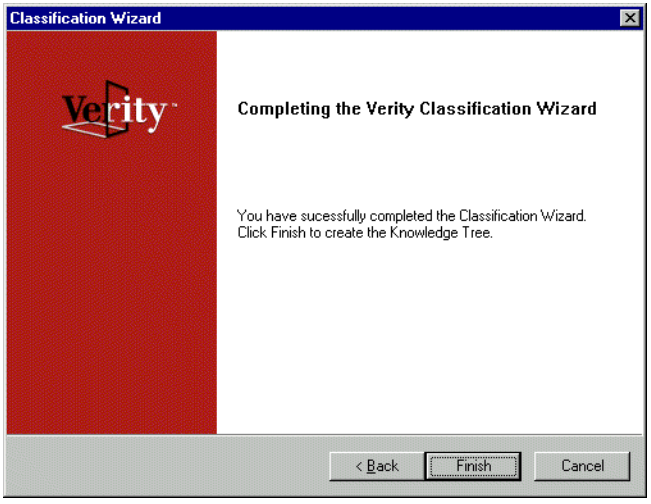
- Clusters—You can generate new categories by grouping similar documents into clusters and create categories based on these clusters.
- Metadata—Generate new categories based on the metadata (fields) of the documents in the collection. For example, you can create a set of new categories where each category contains documents by a certain author.
- File Path or URL—Generate new categories based on path information (either the file path or the URL) of documents in the collection. This creates a hierarchy of categories that mirrors the path hierarchy.

Creating Categories from Document Clusters

1. Click on **Clusters** for the type of Automatic Classification.
2. Click **Next**.



3. In the Clustering Properties dialog, click **Next** to accept the default values.



4. Click **Finish**.

   Intelligent Classifier creates the new Knowledge Tree, topic set, and taxonomy.

Creating and Testing Categories

This section opens a workspace and all related files required to complete this tutorial. If you would like to save the changes you make to the `tutorial_taxonomy.tsw` file for reference at a later date, periodically select **File | Save All** during the tutorial.

Creating Simple Categories

Simple categories are indicated by plain folder icons in the taxonomy tree. They do not contain any documents themselves. They can serve as organizational tools to help structure the taxonomy. You can also modify them so that they do contain documents. You can do this in later parts of this tutorial.

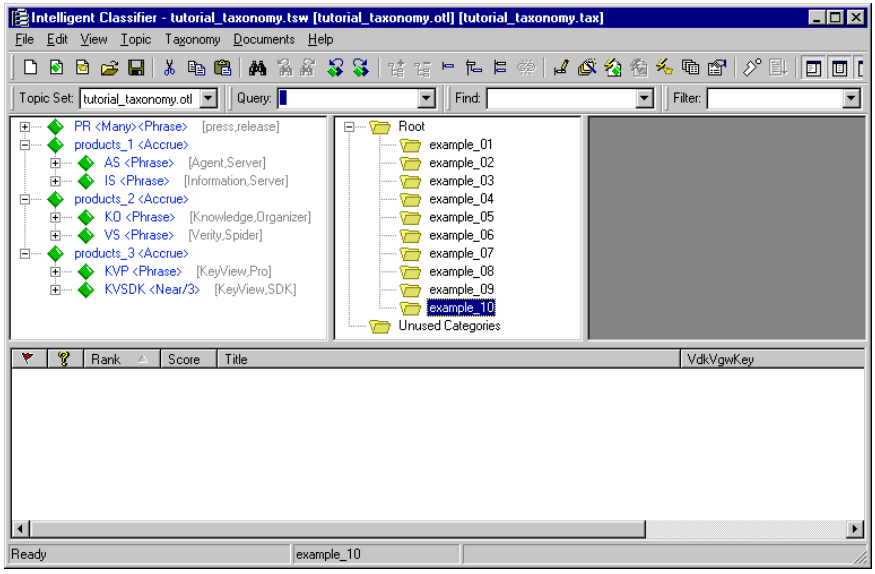
This section shows how to create simple categories.

1. In the Taxonomy Pane, select the **Root** category.

   Root is a predefined category. Root and all its child categories are visible to end users and available for browsing and searching.

   Unused Categories is also a predefined category. Unused Categories and all its child categories are hidden from end users when they use the taxonomy in applications such as Knowledge Organizer. Unused categories provide a place to put categories you do not want to delete but do not want visible to users; this includes categories you are in the process of editing.

2. Select **Edit | Add Child Node**.
3. Intelligent Classifier creates a new child category underneath Root.
4. Name it `example_10`.



Creating Topic-based Categories

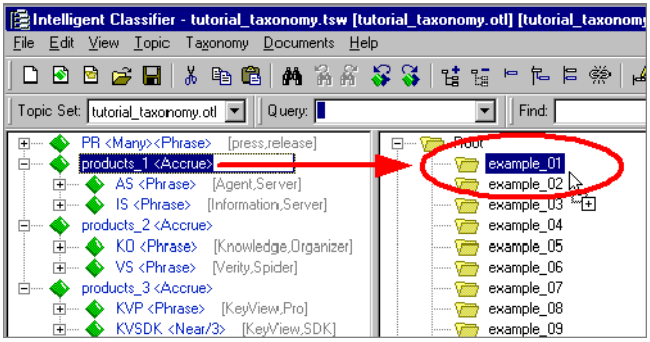
Associating a rule with a category is the most common way to add documents to the category. A rule can be any Verity query. When the rule associated with a category is a topic (a stored query), it is called a topic-based category. Use the **Categories Nodes Properties** dialog box to associate a rule with a category with the following steps:

1. Select a category.
2. Select **Edit | Node Properties**.
3. On the **Main** tab page, check the **Rule** check box.
4. Select a topic from the **Rule** drop-down list that contains all named topics in the active topic set.
5. Click **OK**.

When you browse topic-based categories, you can see the documents ordered by the topic's scores for those documents.

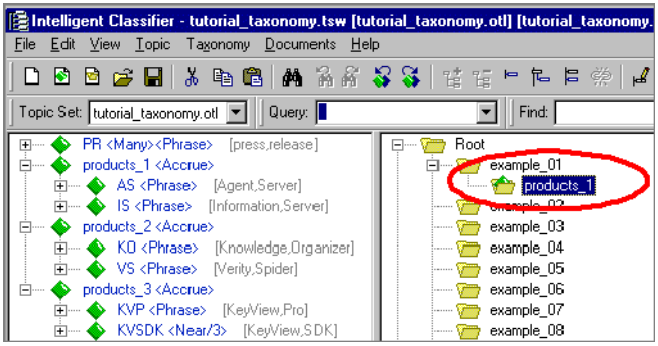
More Ways of Creating Topic-based Categories

1. Select the topic `products_1` from the Topic Pane.
2. Select **Topic | Test Topic** to test that the collection has been added and to view the documents returned.
3. **Drag-and-drop** the topic `products_1` onto the category `example_01` in the Taxonomy Pane.



Note If another node is selected, and you attempt to drag–and–drop a different node, then the original node selected is dropped. To ensure this does not happen, ensure that the node you want to work with has been selected and highlighted in blue.

Intelligent Classifier creates a new category under `example_01` with the same name as the dropped topic.



Notice that the icon for the new category is a folder containing a green diamond. This indicates that the category has a *rule* associated with it. In this case, the rule is the topic `products_1`.

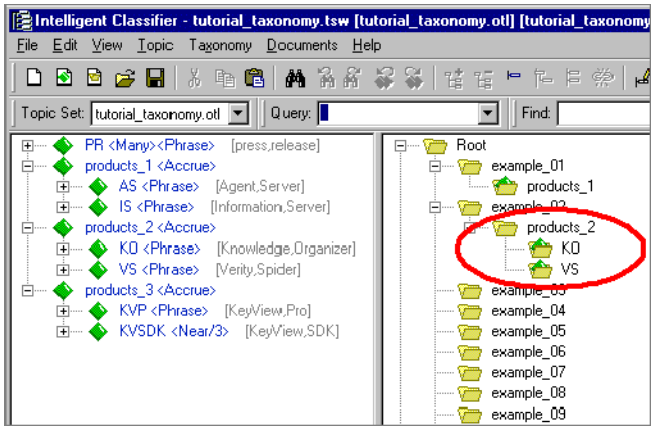
4. Select the *category* `products_1` in the taxonomy pane and choose **Taxonomy | Test Category**.

   Notice that the *category* retrieves the same documents as the *topic*.

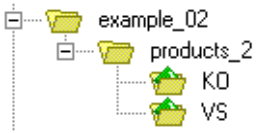
Creating More Complex Hierarchies of Topic-based Categories

This section demonstrates how Intelligent Classifier enables you to easily create more complex hierarchies of topic-based categories.

1. **Ctrl-drag** (hold down the **Ctrl** key *after* you start to drag) the topic `products_2` onto the category `example_02`.

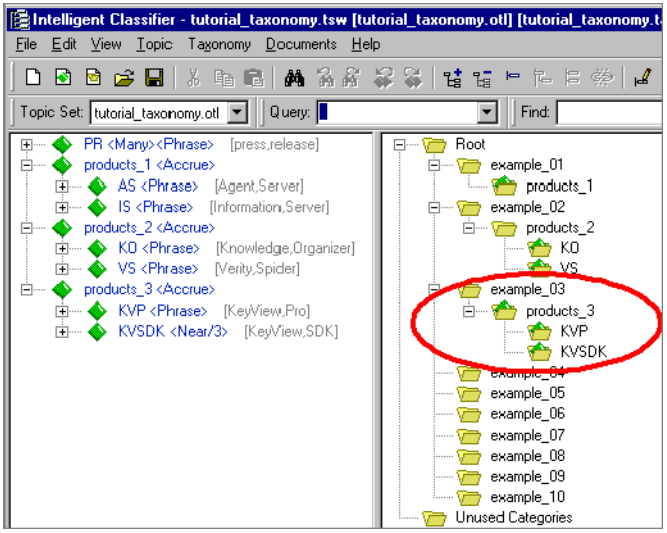


   Notice that, as with the simple drag-and-drop, Intelligent Classifier created a new category with the name of the dropped topic. But in this case Intelligent Classifier also created child categories for the child topics of the dropped topic.

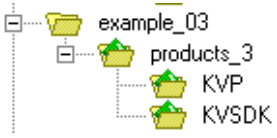


   It also associated the leaf categories (and only the leaf categories) with the corresponding topics. So, in this example, the category `products_2` is not associated with the corresponding topic.

2. Now **Shift-drag** (hold down the **Shift** key *after* you start to drag) the topic `products_3` onto the category `example_03`.



As with the **Ctrl** drag-and-drop, Intelligent Classifier created child categories for the child topics of the dropped topic.



In this case, it associated *all* the newly created child categories (not just the leaf categories) with the corresponding topics. So, in this example, the category `products_3` is associated with the corresponding topic.

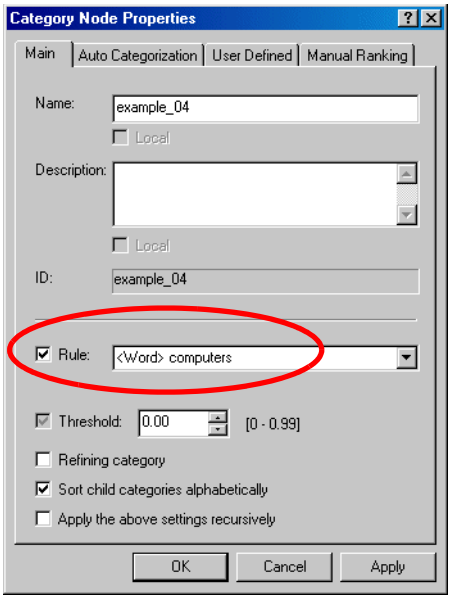
Note If you copy a category to another place in the taxonomy, the copied category references the original category. As a referencing node, the copied category is not allowed to have children. Therefore, only the top-level node are copied; no children are copied.

Creating Query-based Categories

This section shows how to create *query-based categories*. These are like topic-based categories, except that they are based on queries using Verity Query Language (VQL).

1. Select the category `example_04`.
2. Select **Edit | Node Properties**.

The Category Node Properties window opens.



3. On the **Main** tab, select the **Rule** check box, and enter the following query in the Rule field:

   `<Word> computers`

   The Rule field determines what topic or query the category uses.

4. Click **OK** to close the Properties dialog.

   As with the topic-based category, Intelligent Classifier changes the category's icon to a folder containing a green diamond to indicate that this category uses a rule.

5. Right-click on the category and select **Test Category**.

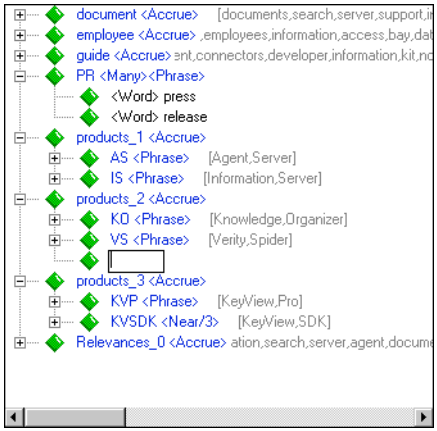
   Notice that Intelligent Classifier retrieves documents containing the word "computers". These are all the documents in the collection matching the query `<Word> computers`.

Updating Categories from Topics

This section discusses how to update a topic-based category if you change the original topic. You add a new child topic called `new` below the topic `products_2`, as shown in the following steps.

1. Select the topic `products_2`.
2. Click **Edit | Add Child Node**.

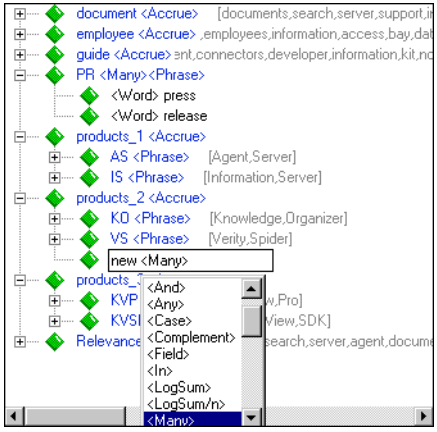
   A new child node appears.



3. Enter the following string text into text box for the node.

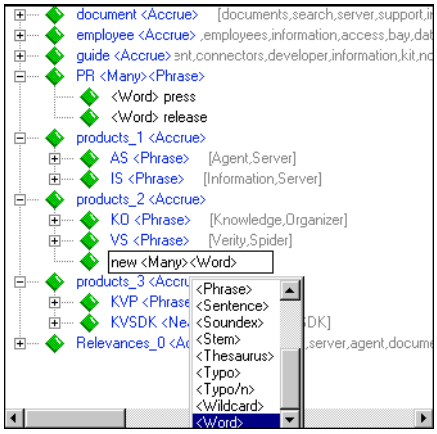
   `new <Many><Word> new`

   When you type the < symbol, a drop–down list appears.



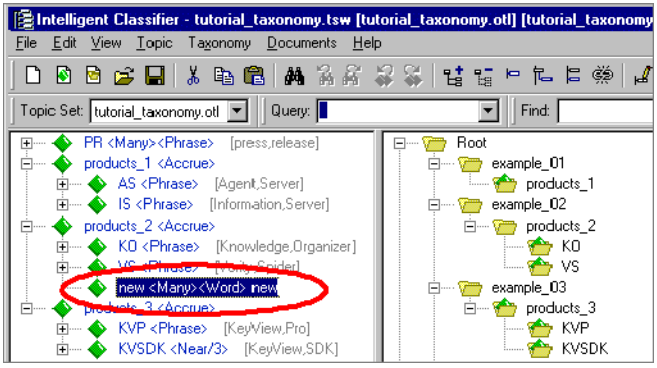
4. Continue entering the text string.

   When you type in the second < another drop–down list appears.

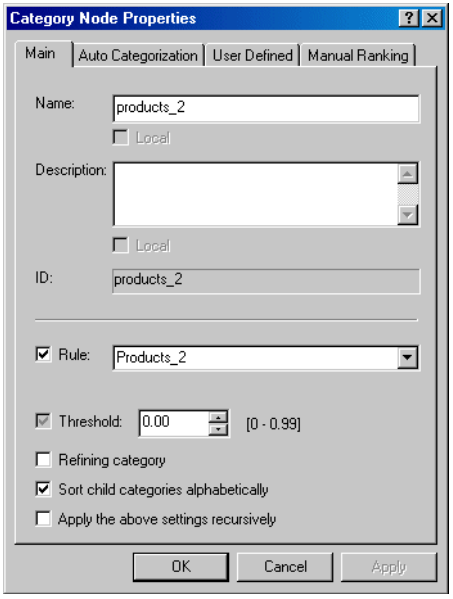


5. Continue entering the text string.
6. Hit **Return** when you have completed entering the text for the child node.

   A new child node appears below products_2.



7. In the taxonomy pane, select the child category `products_2` under `example_2`.
8. Select **Edit | Node Properties**.
9. When the Properties dialog opens click **Rule**. The Rule field determines what topic or query the category uses.



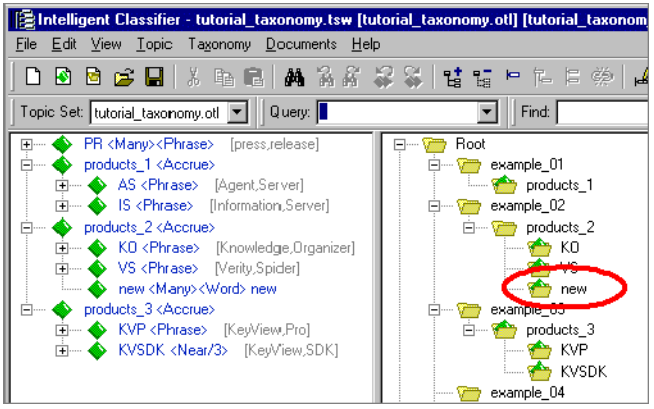
10. Select `products_2` from the drop-down list.

11. Click **OK**.

12. Ensure `products_2` is still selected. If not, select it and select **Taxonomy | Update Category From Topic**.

13. When the dialog appears asking if you want to save changes to the current Topic Set, click **Yes**.

    Intelligent Classifier updates the category to reflect the changes in the original topic. Intelligent Classifier acts as though you repeated the **Shift-drag** or **Ctrl-drag**.



Note If you manually add child categories below `example_02`, Intelligent Classifier does not delete them when it updates the category from the topic. This enables you to preserve manual additions while still being able to easily update the category from the topic.

Manually Adjusting Document Assignment and Ranking

With Intelligent Classifier, you can manually assign to and rank the documents of a category.

Manually Assigning Documents to a Category

This section demonstrates how to manually add documents to a category. You can add documents to both simple categories and rule-based categories. Rule–based categories include both topic- and query-based categories. This provides complete freedom to determine which documents appear in which categories.

1. Test category `example_04`.

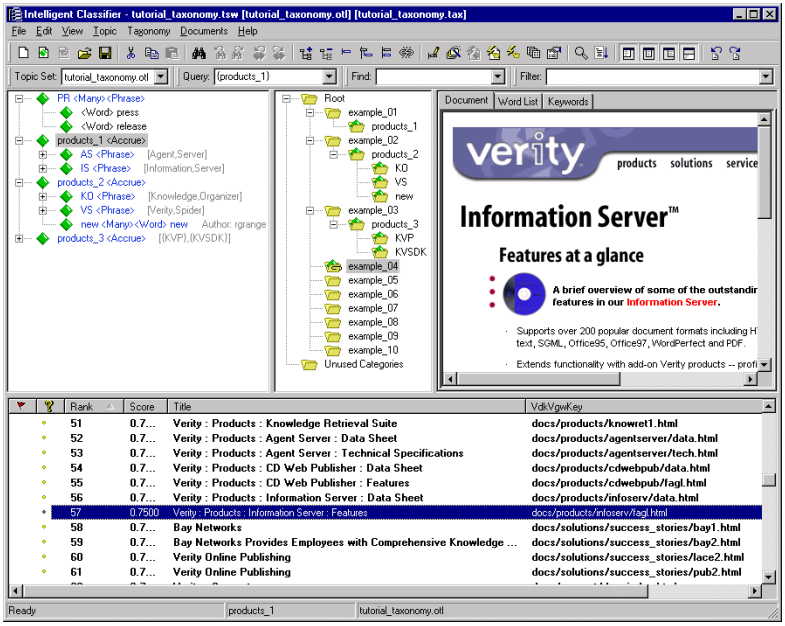
   Notice that it retrieves 7 documents.

2. Test topic `products_1`.

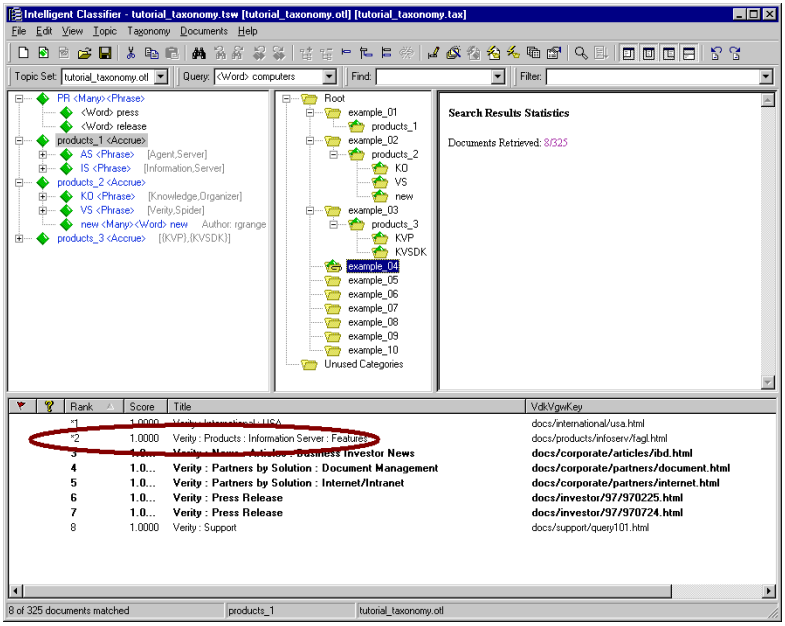
   This retrieves a different set of documents.

3. Drag the document ‘Verity: Products: Information Server: Features’ (it is ranked number 57 so you might have to scroll down to view it) from the Search Results Pane onto the category `example_04`.

   This manually adds it to that category. A category with manually assigned documents is marked by a paperclip symbol to the folder icon.



4. Test category `example_04`. The category now retrieves the added document.



When a category with manually ranked documents Intelligent Classifier displays an asterisk beside the document's number in the Rank column to indicate that it has been manually added. Manually added documents are ranked at the top of the list, immediately below any documents that have been manually ranked.

Manually Removing Documents From a Category

When the Search Results pane's current content results from the testing of a selected category, any document listed can be removed from that category. The following steps demonstrate how to do this:

1. Right-click on a category and select **Test Category**.
2. In the Search Results pane select a document.
3. Select **Taxonomy | Set Manual Ranking**.
4. Select **Exclude from Category**.
5. Click **OK**.

The selected document is now removed from the category.

Removing Manually Assigned Documents

The following steps show how to remove manually ranked or manually excluded document assignments from a category.

1. Select a category with manually assigned documents.
2. Select **Edit | Node Properties**.
3. Select the **Manual Ranking** tab.
4. To remove a manually ranked document, select the document from the **Manually Ranked Documents** list.
5. Click **Remove**.
6. To remove a manually excluded document, select the document from the **Manually Excluded Documents** list.
7. Click **Remove**.
8. Click **OK**.

Adjusting Document Ranking

This section demonstrates how to adjust the ranking of a document in a category. Intelligent Classifier can only adjust ranking for manually assigned documents.

Note Manually assigned documents cannot be ranked lower than non-manually assigned documents. Selecting a non-manually assigned document and changing its ranking creates a manually-assigned document. Therefore, the ranking you specified for this document cannot be lower than non-manually assigned documents.

1. Test category `example_04` to ensure the contents of the Results Pane is current.

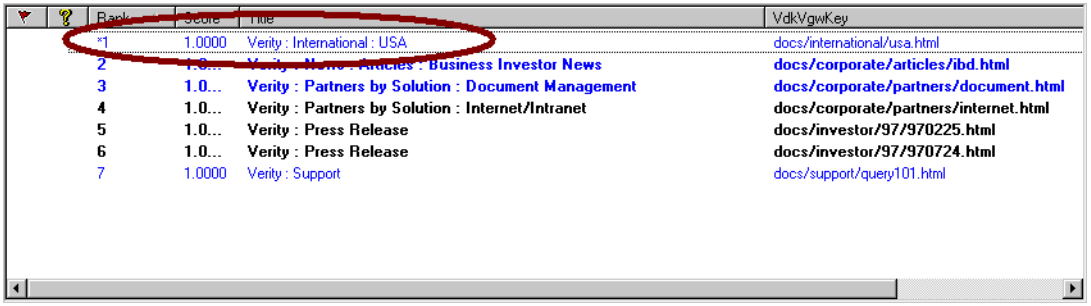
   The document 'Verity: International: USA' is ranked number 4 in the Search Results Pane.

   (The total number of documents in the collection reported by Intelligent Classifier can differ from that shown here depending on what version of Intelligent Classifier you are using.)

2. Select the document 'Verity: International: USA in the Search Results Pane, and select **Taxonomy | Set Manual Ranking**.

   You are unable to change ranking on the document from what appears in the ranking window. This is because this is the first manually ranked document.

3. Click **OK** to close the Manual Ranking dialog.



You can see that the document is now placed at the top of the search results. When an end–user browses this category through Verity Knowledge Organizer, they see this document at the top of the list.

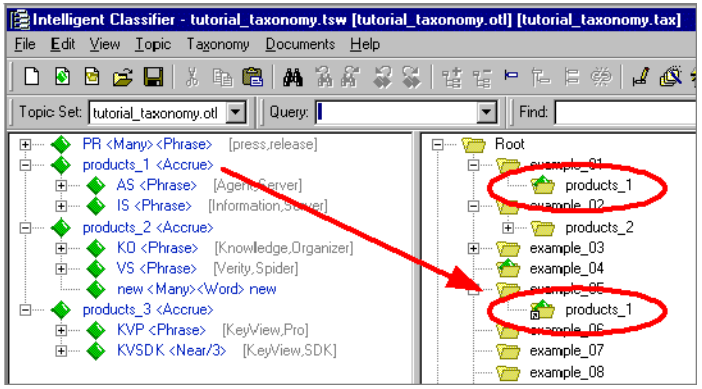
Intelligent Classifier displays an asterisk beside the document's number in the Rank column to indicate that it has been manually ranked.

Using Referenced Categories

A category can occur in more than one place in the taxonomy. When this happens the category copies are said to *reference* the original category. A category that references another is indicated with a small black arrow on the folder icon.

This section demonstrates how to re-use a category as a referenced category in more than one place in a taxonomy.

1. Drag `products_1` onto `example_05`.



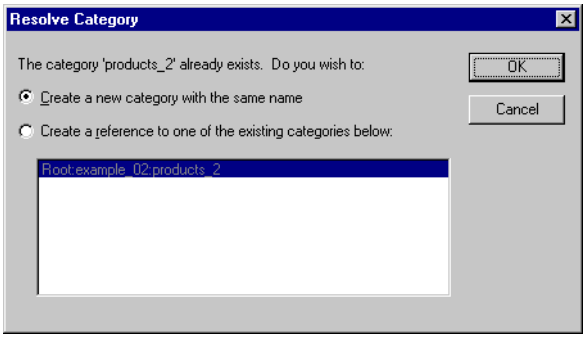
Notice that Intelligent Classifier puts a small arrow on the icon for the new category.

This indicates that this category references the category with the same name that already exists under `example_01`.

When you create a new category by using drag-and-drop or cut-and-paste, if another category already exists with that name, Intelligent Classifier automatically creates a reference to it.

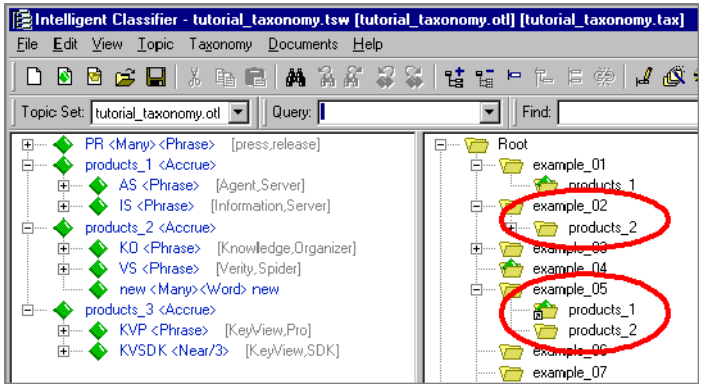
2. Select the category `example_05` and create a child category. Call it `products_2`.

   The Resolve Category dialog appears.



   When you create a new category (by any method except drag-and-drop or cut-and-paste), if a category already exists with that name, you can use Intelligent Classifier to choose whether to create a different category with the same name or create a reference to the existing category. (Unlike topics in topic sets, a taxonomy can contain two distinct categories with the same name.)

3. Choose **Create a new category with the same name**.
4. Click **OK**.



When a category with manually ranked documents is tested, Intelligent Classifier creates an new category. It has the same name, `products_2`, as the category under `example_02`, but is completely independent. Notice that Intelligent Classifier uses a plain folder icon for this category.

To break a reference link for a referenced category:

1. Select a referenced category.
2. Select **Edit | Break Link**.

Creating Refining Categories

Normally, topic-based or query-based categories execute their rules against all the documents in a collection. You can create *refining categories* to contain a subset of documents from a parent category. This enables you to easily limit the set of documents that can be distributed into certain categories.

This section demonstrates how to create *refining categories*.

1. Drag the topic `PR` to the category `example_06`.

   This creates a new category `PR` under `example_06`.
2. Drag the topic `products_1` to the new category `PR`.
3. Right-click on the `products_1` category.
4. Select **Break Reference**.

5. Select the child category `products_1` under `PR` and choose **Edit** | **Node Properties**.

   The Properties dialog opens.

6. Check **Refining category**.

7. Click **OK** to close the dialog.

   Notice that Intelligent Classifier now places a checkmark on the category's icon. This indicates that the category is now a refining category.

8. Test the category `products_1`.

   Notice that the documents retrieved are those that contain "press release" *and* one of the phrases selected by `products_1` (for example, "Agent Server" or "Information Server" or in this example "What's New").

Turning Off Refining Categories in a Reference Category

A category must be either a refining category, or a reference category. If you move a refining category to become a child of a reference category, you must break the reference link to the former refining category.

1. Right-click the new child category and select **Break Reference**.

2. Right-click the child category again. Uncheck **Refining Category**.

Updating Properties for Multiple Categories

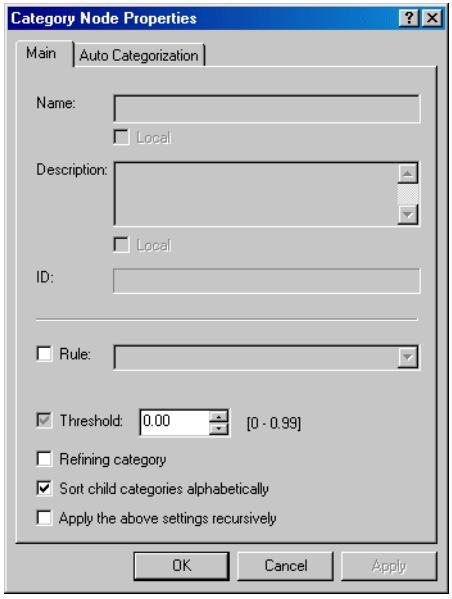
You can select and update the properties of multiple category nodes all at once. Properties that you can modify in this way are **Rule**, **Threshold**, **Sort child categories alphabetically**, and **Refining category** within the **Properties** dialog box. Changing any of the properties on this page updates all the selected nodes. This is useful when clearing the properties for multiple category nodes that have been imported.

This section shows how to update multiple categories.

1. Select the categories `example_01`, `example_02`.

2. Choose **Edit | Node Properties.**

   The Category Node Properties dialog opens.



3. Check **Refining category** on the Main tab. Click **OK**.

   Notice that Intelligent Classifier has placed a checkmark on each selected category's icon. This indicates that all selected categories are now refining categories.

Recursively Updating Child Nodes

The **Apply the above settings recursively** check box updates the properties of the selected nodes, and any of their child nodes.

1. Select the topic `example_02`.
2. Choose **Edit | Node Properties.**
3. Check **Refining category**.
4. Check **Apply the above settings recursively**. Click **OK**.

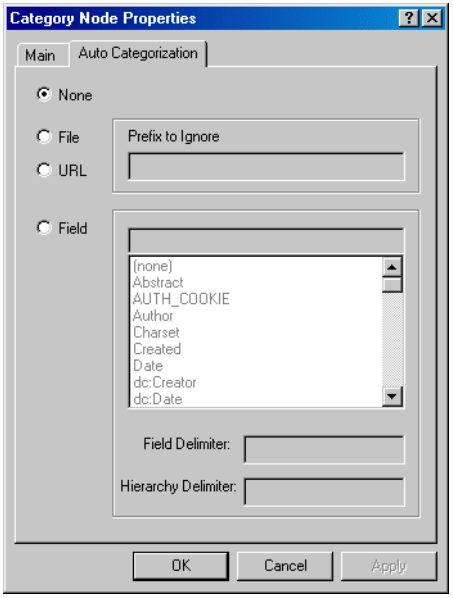
   Intelligent Classifier places a checkmark on each descendant category's icon for example_02, indicating that the descendant categories are also refining categories.

Note When several items are selected, the options on the main page of the node properties dialog are checked or grayed if not all categories have the value set.

Updating Automatic Categorization

You can update multiple automatic categorization categories.

1. Select the categories `example_01`, `example_02`.
2. Choose **Edit | Node Properties**. The Category Node Properties dialog opens.



3. Click on the **Auto Categorization** tab.
4. You can select the following:
   - None—removes properties and any automatically-generated child categories from the selected categories.
   - File—enables the Prefix to Ignore text box.
   - URL—enables the Prefix to Ignore text box.
   - Field—enables the drop-down list for you to select fields. You can also type in field and hierarchy delimiters.
5. Click **OK**.

Automatic categorization with multiple categories selected is not performed if you have one or more reference categories selected, or if you have an automatic categorization parent and one or more of its children simultaneously selected.

Creating Categories Automatically

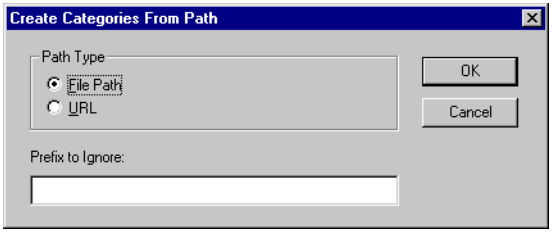
Intelligent Classifier's powerful automatic categorization feature enables you to create categories automatically, based on a number of possible criteria.

Automatic Categorization By Path

This section demonstrates how Intelligent Classifier can create categories based on the documents' file paths. This provides an easy way to create categories that mirror the directory structure of a collection. This is especially useful if your directory structure provides useful groupings of documents.

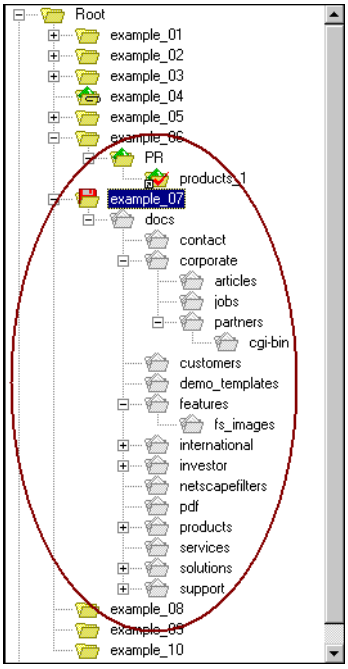
1. Select category `example_07`.
2. From the Intelligent Classifier menu bar, select **Taxonomy | Automatically Create | Categories From Path**.

   The **Create Categories From Path** dialog opens.



3. Select the **File Path** option.
4. Click **OK** to close the dialog.

   Intelligent Classifier automatically creates new child categories, with each level in the hierarchy corresponding to a segment in the document's file paths.



Notice that the icons for the parent of the new categories show a folder enclosing a floppy disk. This indicates it is file based.

The new categories are shown in gray. This indicates that they are generated when the taxonomy is used in applications such as Knowledge Organizer and are based on the collection's documents. As the documents in the collection change, these categories automatically update to reflect these documents. Intelligent Classifier shows a preview of what the categories are *based on the current state of the attached collections*. They can be treated just like regular categories, except that they cannot be deleted.

When your collection is based on a Web site, you can create categories automatically based on the documents' URLs instead of paths.

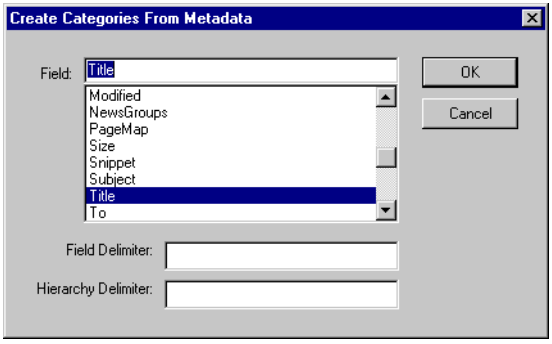
When the parent category is rule-based, Intelligent Classifier creates new categories only for the documents contained in the parent. When the parent is a simple category (that is, one that is not rule-based), Intelligent Classifier uses all the documents in a collection.

Automatic Categorization Using Metadata

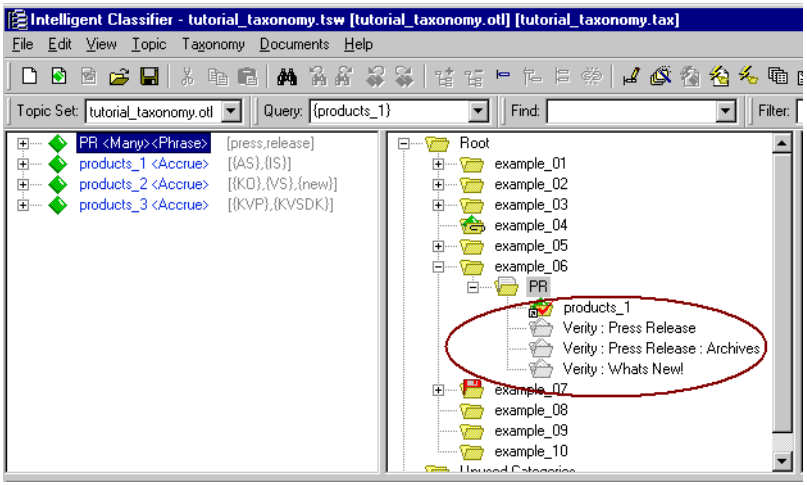
This section demonstrates how to automatically create categories based on the collection's *fields*.

1. Select the category `PR` under `example_06`.
2. From the Intelligent Classifier Main Menu, select **Taxonomy | Automatically Create | Categories From Metadata**.

   The **Create Categories From Metadata** dialog opens.



3. Select **Title** as the value for the **Field** text entry box.
4. Click **OK** to close the dialog. Intelligent Classifier creates categories based on the document titles.



Because the parent category (PR) is rule-based, Intelligent Classifier creates categories only for the documents contained in the parent category.

You can enter a string in the Field Delimiter box or Hierarchy Delimiter box. The entire string is used as a separator for field or hierarchy when Intelligent Classifier parses the metafield.

Removing Automatic Categorization by Path or Metadata

This section demonstrates how to remove automatic categorization.

1. Select the root node of an automatically generated taxonomy subtree.
2. Select **Edit | Node Properties**.
3. Select the Auto Categorization page.
4. Select **None**.
5. Click **OK**.

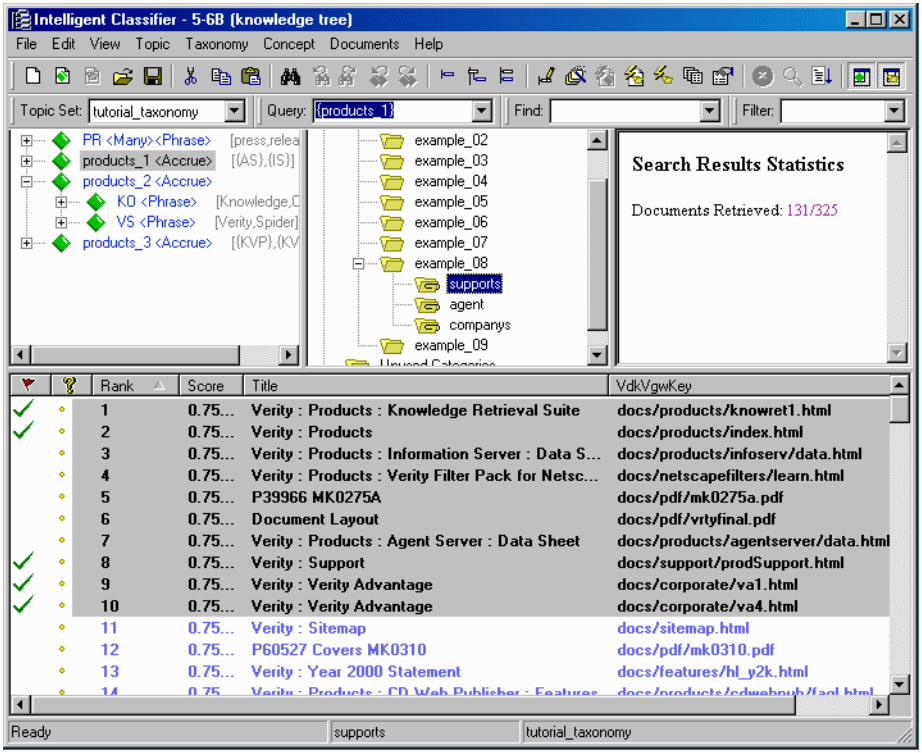
Note Rules added during the automatic generation or manually are not automatically removed when you set the auto categorization property to **None**. You need to manually remove these as well.

Automatic Categorization from Document Clusters

This section demonstrates how Intelligent Classifier makes it easy to select a set of documents and then automatically group similar documents into clusters and create categories from these clusters.

1. Test the topic `products_1` to ensure that the documents for this topic are current in the Results Pane. Documents in the result list from this topic are used as a base for the new categories.
2. Select (by **Shift**-clicking) the first ten documents in the Search Results pane.
3. Select category `example_08`.
4. From the Main Menu, select **Taxonomy | Automatically Create | Categories From Document Clusters**.

   Intelligent Classifier creates three new child categories underneath the `example_08` and associates each with a set of documents.



If you test the child categories under `example_08`, Intelligent Classifier retrieves documents similar to the ones you selected.

Categories generated by document clustering are actually normal category nodes with manually assigned documents. This is different from categories generated automatically by path or metadata.

Importing and Exporting Taxonomy Files

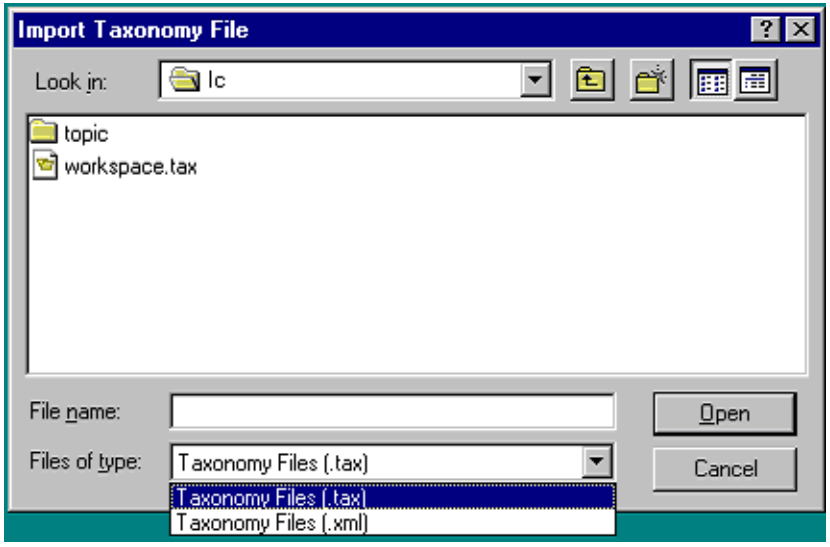
Taxonomy files can be saved to and imported from two types of file formats: TAX and XML. For information about these formats, see "Using Taxonomy Definition Files."

Importing Taxonomy Files

This section shows how to import a taxonomy file in Intelligent Classifier.

1. To import a taxonomy file, select **Taxonomy | Import | Taxonomy**.

   The Import Taxonomy File Dialog appears.



2. Select the type of file you wish to open from the **Files of type** drop-down list box.
3. Enter the filename in the **File name** edit box, or select the file from the file list pane.
4. Click **Open**.

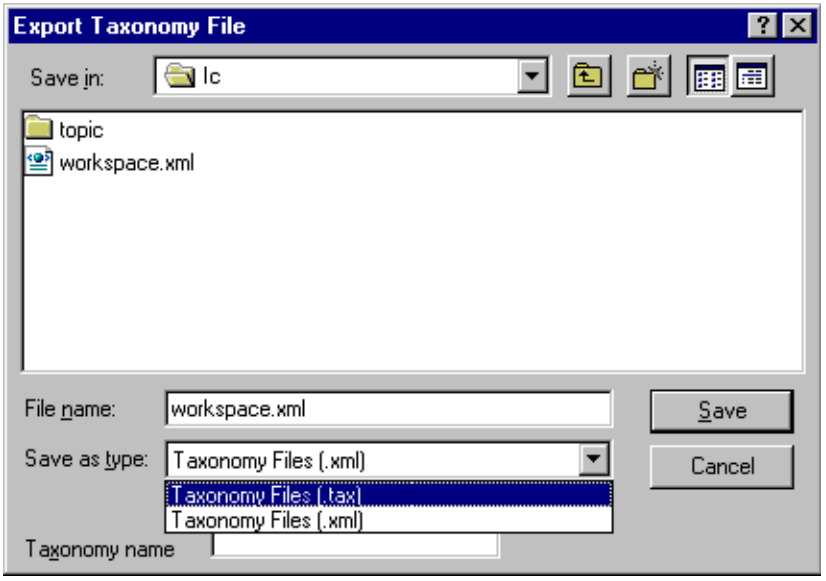
   The taxonomy displays in the Taxonomy pane of Intelligent Classifier.

Exporting Taxonomy Files

This section shows how to export a taxonomy file in Intelligent Classifier.

1. Select **Taxonomy | Export Taxonomy**.

   The Export Taxonomy File dialog box appears.



2. Select the type of file you wish to export to from the **Save as type** drop-down list box.
3. Enter the filename in the **File name** edit box, or select and existing file from the file list pane.
4. Click **Save** to export the file.

When you choose to export to an XML format taxonomy file, you can specify a name for it in the **Taxonomy name** edit box that is then saved as the value of the "Name" attribute of the "Taxonomy" element in the XML taxonomy file. The **Taxonomy name** default value is the root node name.

5

Thematic Mapping

This chapter describes Intelligent Classifier's *thematic mapping* functionality. It contains the following sections:

Introduction to Thematic Mapping

Thematic mapping is a powerful classification feature that is unique to Intelligent Classifier. It automatically extracts key terms (nouns or *noun phrases*) contained in a set of documents and organizes them in a hierarchy – a concept tree. Concept trees provide:

- Easy visualization of subject areas in the document set
- Document-search based on key concepts in the document set
- An automatic (or semi-automatic) method of creating and populating a taxonomy

Thematic mapping discovers contextual concepts (terms that are semantically related based on their distributional patterns in the document set). These contextual concepts might not match the domain-level concepts the user has in mind. Also, thematic Mapping only generates one possible concept hierarchy out of the many possible concept hierarchies for the document set. The concept hierarchy generated might not completely match the one that the user has in mind.

Collection Requirements

Thematic mapping works only on collections built with indexed nouns and noun phrases. For information on building this type of collection, see the chapter on index tuning in the *Verity Collection Reference*. The default filter for PDF documents in `style.uni` is `flt_kv`, which allows you to index nouns and noun phrases in PDF documents.

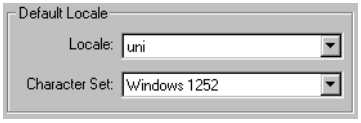
It is recommended that the collection contain at least 2048 documents.

The collection needs to be optimized with a spanning word list. This can be done by letting Intelligent Classifier build the span word list when it finds the collection does not have a spanning word list for certain operations, like LRC topic creation and thematic mapping. You can also do the optimization on the command line before starting Intelligent Classifier:

```
mkvdk -collection <collection_name> - optimize spanword -locale
<locale_name>
```

See the *Verity Collection Reference* for more details.

To extract noun and noun phrases properly, Intelligent Classifier's session locale must be consistent with the collection's locale and character set, which means that you must use the same locale as the collection's locale when you create the workspace. Use **View | Options** to select the locale for creating a new workspace.



System Requirements

The thematic mapping processes require a substantial amount of memory space, and possibly a substantial amount of disk space. The amount of memory space and disk space required depends on the:

- Number of key terms extracted from the collections (see the next section for details)
- Size of the collections (the number of documents in the collections)

When the amount of memory space available is insufficient, a temporary cache file is created for storing the intermediate data. The cache file is created under the path specified by `%TMP%` (or `%TEMP%` if `%TMP%` is not defined) environment (user) variable. If neither `%TMP%` nor `%TEMP%` is defined, the cache file is created in Windows' default directory `C:\Winnt`. By default, both variables point to `%USERPROFILE%\Local Settings\Temp`. The process will fail if the disk drive on which the cache file is located runs out of space. When the collections to be processed contain a large number of documents, ensure that there is a substantial amount of disk space on the drive on which the cache file is located. The cache file is removed after the process finishes.

What is a Concept?

In the context of Intelligent Classifier's thematic mapping, a concept is defined by a term or a set of terms. A term defines a base concept, and a set of terms together defines a high-level concept. For example, "car," "truck," "motorcycle" are terms that define their corresponding base concepts, and together, the set of these three terms can define the higher level concept "motor vehicle."

What is a Concept Tree?

Two or more high-level concepts together define another high-level concept. For example, the concept of *motor vehicle* (mentioned above) and the concept *non-motor vehicle* defined by the set of terms "carriage," "trailer," and "railcar" define the high-level concept *vehicle*. The base concepts "carriage," "trailer," and "railcar" are viewed as children of the high-level concept *non-motor vehicle*, which is viewed as the parent of the base concepts.

Similarly, the concept *vehicle*, together with other concepts, can define yet another high-level concept. Concepts of different abstraction levels can be organized in a hierarchical structure in this manner. This hierarchy of concepts is called a *concept tree*.

Nodes in a Concept Tree

There are two types of nodes in a concept tree: nodes that correspond to base concepts are called *term nodes*; nodes that correspond to high-level concepts are called *concept nodes*.

Creating a Concept Tree

This section describes how to create and manage a concept tree.

Setting the Concept Extraction Criteria

The first step of the concept tree creation process is to extract the key terms contained in the document set. The most important noun and noun phrases contained in the document set are taken as the key terms. Intelligent Classifier uses one of three possible criteria to measure the importance of a term:

- Term Frequency—the total number of times a term occurs in the document set
- Document Frequency—the number of documents in the set in which the term occurs
- TFIDF—Term Frequency divided by the Document Frequency.

The default criterion is Term Frequency.

To start creating a concept tree, start Intelligent Classifier. The Knowledge Tree wizard is displayed automatically. You can use the wizard or open a new workspace.

Select The Criteria For Measuring Term Importance

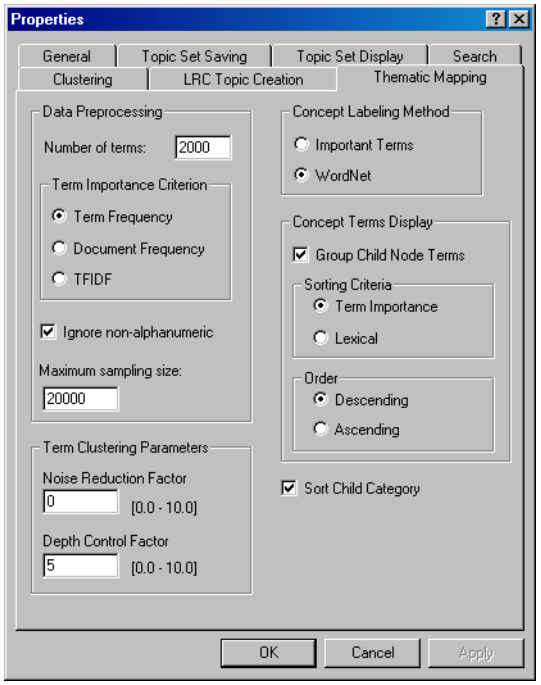
1. Select **File | Properties**.
2. Click the **Thematic Mapping** tab.
3. Under Data Preprocessing, select the desired criterion under Term Importance Criterion.

4. Click **OK**.

Determine the Number of Terms to Extract as Key Terms

1. Select **File | Properties** and click the **Thematic Mapping** tab.
2. In the Data Preprocessing section, type the number you want in the **Number of Terms** field and click **OK**.

   The default value is 2000.



Note While the thematic mapping process can handle any number of key terms, the process can become very slow when a very large value is used. See "The thematic mapping process is slow" in "Thematic Mapping from the Command Line" for details.

Note The concept tree generated can contain fewer terms than specified, as some of the terms selected in this process can be found unsuitable to form part of the concept tree.

Setting other Intelligent Classifier Properties

Increase the number of documents returned in the results list.

1. Select **File | Properties**.
2. Click the **Search** tab and type a value greater than 2048 in the **Maximum Number of Documents** field.
3. Click **OK**.

Building a Concept Tree

1. Ensure at least one collection is currently open.
2. Select **Concept | Create Concept Tree From**.
3. Select one of the following options:
   - All Documents in Collection
   - Documents in Search Results pane (only available after a search has been conducted)
   - Selected Documents (only available when some of the documents listed in the search results pane have been selected)

   The generated concept tree will appear in the concept tree pane.

You can view the progress in the progress bar at the bottom of the page.

Concept Tree Management

The following sections describe how to open, save, and close your concept trees.

Saving a New Concept Tree

A generated concept tree can be saved in either `.ct` or `.xml` format. The `.ct` format uses a pair of `.ct` and `.trm` files to represent a concept tree. The XML format is defined by taxonomy.dtd, which is located in `install_dir\common`. For information about the taxonomy.dtd, see "Using Taxonomy Definition Files."

To save a concept tree:

1. Select **Concept | Save Concept Tree As**.

2. Select the file format in **Save as type**.
3. In **File name**, enter the name and location of the concept tree file you want to create.
4. Click **Save**.

If `.ct` format is selected, a matching `.trm` file is also created in the same location as the `.ct` file. The `.ct` and `.trm` files should be kept in the same directory. Otherwise, Intelligent Classifier will not be able to open the saved concept tree. Also, when moving or copying a `.ct` file, remember to move or copy the matching `.trm` file.

If `.xml` format is used, one XML file is created.

Opening a Saved Concept Tree

1. Select **Concept | Open Concept Tree**.
2. Select the file format in **Save as type**.
3. In **File name**, enter the name and location of the concept tree file you want to open (the matching `.trm` should be in the same location).
4. Click **Open**.

Closing a Concept Tree

1. Select **Concept | Close Concept Tree**.

Advanced features

The following sections describe advance features you can use when creating a concept tree.

Concept Labeling

A label is generated for each concept node in the concept tree. When the Important Terms option is selected, the two most important terms of the concept, according to the Term Importance Criteria selected, are used to form the label.

By default, Intelligent Classifier uses a labeling method that labels a concept with two terms that are most descriptive of the concept as a whole, determined using the WordNet Lexical Database. For example, for the concept that contains the terms “milk”, “soda”, “bottled water” and “beer”, the terms “beverage” and “drinks” are more descriptive of the concept as a whole, compared to any individual term in the concept, and you can use them to form the label of the concept.

Selecting a Concept Labeling Method

1. Select **File | Properties**.
2. Click the **Thematic Mapping** Tab.
3. In the **Concept Labeling Method** section, select WordNet or Important Terms.
4. Click **OK**.

The concept labeling process uses two stopword lists. You can modify the behavior of the concept labeling process and the labels generated for the concept by editing these two stopword lists. The two lists are stored in the `Labels.stp` and `Terms.stp` files, which are located at `$INSTALL\_nti40\concept\labeling\stopword`. Terms found in `Terms.stp` are ignored in the labeling process even when they appear in a concept. Terms found in `Labels.stp` will not be used as part of the concept labels.

Noise Reduction

The quality of the generated concept tree depends on the quantity and quality of the documents. When a small set of documents is used (for example, fewer than a few thousand documents), or the documents contain a large amount of noise (for example, web pages containing a large number of templates not related to the main theme of the page), the quality of the generated concept tree might not be satisfactory.

When a small or *noisy* document set is used, the quality of the generated concept tree can be improved by using a higher Noise Reduction Factor.

Setting the Noise Reduction Factor

1. Select **File | Properties**.
2. Click the **Thematic Mapping** Tab.
3. In the **Term Clustering Parameters** section, type in the desired value in the edit box marked **Noise Reduction Factor**.
4. Click **Apply**.
5. Click **OK**.

Note The process of noise-filtering usually removes a small amount of useful data as a by-product. Therefore, setting this parameter too high can result in deterioration in the results. Adjust this parameter gradually and observe the changes quality with respect to the changes to determine the optimal setting.

Hierarchy Depth Control

You can control the depth (and flatness) of the generated concept tree through the Depth Control Factor parameter. Increasing the value of this parameter may increase the depth of the concept tree generated, and vice versa.

Setting the Depth Control Factor

1. Select **File | Properties**.
2. Click the **Thematic Mapping** tab.
3. In the **Term Clustering Parameters** section, type the value in the **Depth Control Factor** field.
4. Click **Apply**.
5. Click **OK**.

Note Altering the value of this parameter does not always result in the change of the depth of the tree. The depth, flatness, and structure of the generated tree depends on both the semantic closeness between the concepts/subtrees in the tree, and the value of this parameter. Increasing, or decreasing, this value by a small amount may have no effect on the depth of the concept tree generated when the concepts in the tree are very far apart, or very close semantically. If you do not see any changes to the depth of the tree after altering this value, try increasing or decreasing it by a larger amount.

However, if the tree already has its maximum or minimum depth allowable by the semantic distance between the concepts, increasing or decreasing this value will not have any effect. For example, when the generated concept tree contains some unrelated (that is, semantically very distant) top level concepts/subtrees, increasing the depth value by any amount will not force these unrelated concepts/subtrees to be grouped together. Therefore, increasing the value of the parameter will not affect the depth.

Similarly, when concepts under each of the top level subtrees are very closely related, decreasing the value of the parameter may not affect the depth of the tree.

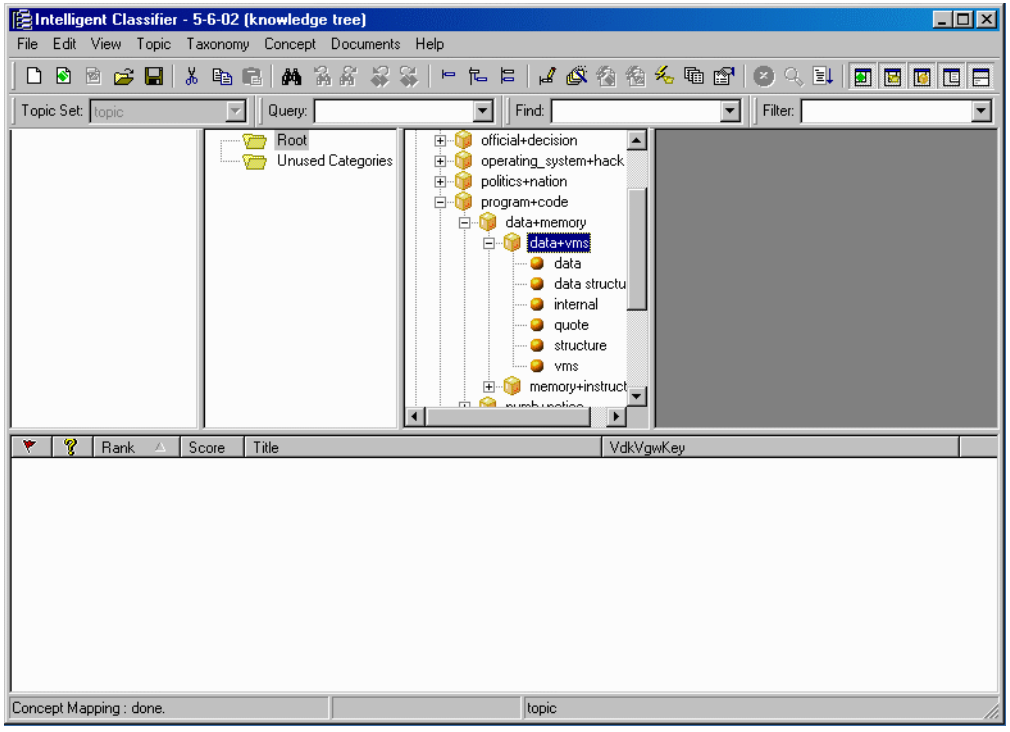
Browsing a Concept Tree

This section describes how to browse the contents of a concept tree using Intelligent Classifier's graphical user interface.

Representation of a Concept Tree

A concept tree is displayed in the concept tree pane. Term nodes (base concept nodes) are represented by solid spheres and concept nodes (higher level concept nodes) are represented by cubes.

Figure 5-1 Concept Tree pane



Expanding and Collapsing a Concept Node

To expand a node in the concept tree:

- Click the + sign.

OR

- Right-click a node and select **Expand**.

To collapse an expanded node in the concept tree:

- Click the - sign.

OR

- Right-click on a node and select **Collapse**.

To expand or collapse multiple nodes:

1. Select the nodes you want to expand or collapse.
2. Right-click the nodes.
3. Select **Expand** or **Collapse**.

Order of the Child Nodes

The term nodes and concept nodes under a particular concept node are sorted in alphabetical order by default.

To disable this feature for a particular concept node:

1. Right-click the node.
2. Select **Sort Child Nodes**. The check mark in front of the command name disappears.

Follow the same steps to enable the feature.

Displaying Terms in a Concept

To display all the terms/base concepts under a particular concept node in the concept tree:

1. Select a node.
2. Select **Concept | Display Terms in Concept**.

OR

1. Right-click the node.
2. Select **Display Terms in Concept**.

You can determine the way the terms are displayed by setting the corresponding options in the properties page:

1. Select **File | Properties**.
2. Click the **Thematic Mapping** tab.

The term display options are shown in the Concept Terms Display section.

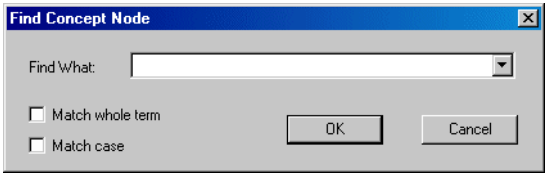
- To display terms under different child nodes in groups represented by three different icons (colors and shapes), select the **Group Child Node Terms** option. The three icons will alternate in the vertical view.
- To display all terms under the current node in a single group, uncheck the box.
- To sort the terms by Term Importance, select **Term Importance** in the **Sort Criteria** section.
- To sort the terms in Lexical order, select **Lexical**.
- To sort the terms in Ascending Order, select **Ascending in the Order** section**.**
- To sort the terms in Descending Order, select **Descending**.

Searching for a Concept

To search for a concept (base concept or higher level concept) in the concept tree:

1. Right-click in the concept tree pane.
2. Select **Find**.

   This brings up the **Find Concept Node** dialog.



3. Type the name of the concept you want to search for in the **Find What** field.

Check **Match whole term** to find only whole words. Leaving this unchecked will find words where the stem matches within the word.

Check **Match case** when you want the search to be case sensitive. Leaving this unchecked will perform a case-insensitive search.

4. Click **OK**.

You can also select a concept in the Find bar when the Concept pane is the current focus pane. The pull-down list contains all the concepts.

The matched concept will be highlighted with the path to the matched node expanded. When multiple nodes are matched, you can use the find-next or find-previous button to navigate through the concept pane.

Conceptual Search

You can search documents in one or more collection(s) based on a concept in a concept tree. You can also create a Verity Topic from a concept.

Conducting a Conceptual Search

Conducting a Search Based on a Concept in the Concept Tree (Conceptual Search):

- Double-click the corresponding node in the concept tree.

OR

- Right-click the node and select **Test Concept.**

You can set the Threshold value for conceptual searches based on a node.

Setting the Threshold Value

1. Right-click a node.
2. Select **Properties**.
3. Adjust the Threshold value.
4. Click **OK**.

Creating a Topic from a Concept

Creating a Topic from a Concept Node in the Concept Tree

1. Right-click the node.
2. Select **Create Topic From Concept**.

After a Topic is created from a concept node, a green diamond appears on the box representing the concept. The diamond signifies that the generated Topic is associated with the concept node. When there is a rule associated with a concept node, a conceptual search based on the node uses the associated rule for searching. The rule associated with a node, and the results obtained by conducting a search on the node, remain the same even if the content (the children and descendants) of the concept node has been changed.

Disassociating a Topic from a Node

1. Right-click the node.
2. Select **Properties**.
3. De-select the **Rule** option box.
4. Click **Apply**.
5. Click **OK**.

You can also choose to associate any existing Topic to a node.

Associating a Topic in the Topic Set with a Node:

1. Right-click the node.
2. Select **Properties** and click **Rule**.
3. In the **Rule** drop-down list, select a topic you want to associate with the node.
4. Click **Apply**.
5. Click **OK**.

Creating Topics from a Concept Node and its Children

You can create topics for a concept node and each of the concept nodes under it using the following procedure.

1. Right-click the node.
2. Select **Create Topic From Concept Recursively**.
3. Each of the concept nodes will have a matching top-level topic.

Editing a Concept Tree

This section describes how to edit a concept tree using Intelligent Classifier.

Renaming a Node

1. Right-click the node.
2. Select **Rename**.
3. Edit the name of the node.
4. Click the empty space of the pane or press **Enter** after you edit the name.

You can also enter the editing mode by clicking a selected node.

Deleting a Node in the Concept Tree

1. Right-click the node.
2. Select **Delete** and click **Yes** to confirm.

Deleting Multiple Nodes in the Concept Tree

1. Select the nodes you want to delete.
2. Right-click.
3. Select **Delete** and click **Yes** to confirm.

You can also use the **Delete** key to delete a selected concept node(s).

Moving a Node to Another Location in the Tree

1. Click the node.
2. Drag it to the desired location and drop.

Moving Multiple Nodes to a New Location

1. Select the nodes you want to move.
2. Drag them to the desired location and drop.

Move one or more Nodes to a New Location Using the Cut and Paste Commands

1. Select the node(s) you want to move.
2. Right-click one of the highlighted node(s).
3. Select **Cut**.
4. Right-click the node you want the cut nodes pasted to.

Note You cannot paste any node under a term node.

5. Select **Paste**.

Making a Copy of a Node

1. Right-click the node.
2. Select **Copy**.
3. Right-click the node you want the new node copied to.
4. Select **Paste**.

Copying Multiple Nodes

1. Select the nodes you want to copy.
2. Right-click one of the selected nodes.
3. Select **Copy**.
4. Right-click the node you want the cut nodes pasted to.

Note You cannot paste any node under a term node.

5. Select **Paste**.

Adding a Term Node as a Child of a Concept Node

1. Right-click the concept node.
2. Select **Add Term**.
3. Type in the term and its importance value in the dialog box.
4. Click **OK**.

Adding a Concept Node as a Child of Another Concept Node

1. Right-click the parent node.
2. Select **Add Child**.
3. Type in the name of the new node.
4. Press **Enter** to add another node, or click the empty space of the pane to finish.

Adding a Concept Node as a Sibling of Another Node

1. Right-click the node.

2. Select **Add Sibling**.
3. Type in the name of the new node.
4. Press **Enter** to add another node, or click the empty space of the pane to finish.

Building a Taxonomy Using a Concept Tree

A concept tree derived from a document set provides easy visualization of the key topics. You can use the concept tree as the basis for creating a taxonomy or parts of a taxonomy of documents of similar content. In most cases, the concept tree does not have the exact structure of the ideal taxonomy you want; however, you can use the concept tree to reduce the effort needed to create the taxonomy. You can edit the concept tree and then derive the final taxonomy from the concept tree, or derive a preliminary taxonomy from an unedited concept tree and then edit the derived taxonomy. You can also use selected parts of a concept tree to form parts of a taxonomy or to extend an existing taxonomy.

Deriving a Taxonomy from a Concept Tree

You can derive a taxonomy directly from a concept tree. The derived taxonomy will have the same hierarchical structure as the concept tree from which it is derived, except for the term nodes. Any topics associated the individual concept nodes will also be associated with the corresponding categories in the derived taxonomy.

Deriving a Taxonomy from a Concept Tree

1. Select **Taxonomy | New Taxonomy**.
2. Select the name and location of the new taxonomy you want to create in the pop-up dialog box.
3. Click **OK**. An empty taxonomy appears in the taxonomy pane.
4. Select all concept nodes under the root node of the concept tree.
5. Drag it onto the root category of the empty taxonomy.

Creating a Taxonomy using a Concept Tree

You can create parts of a taxonomy, or extend an existing taxonomy, using a concept tree, or some part(s) of it.

Creating Parts of a Taxonomy Using a Concept Tree

1. Open a new taxonomy or an existing taxonomy.
2. Select the part(s) of the concept tree you want to convert into part of your taxonomy.
3. Drag your selection to the desired location of the taxonomy.

Repeat these steps to extend the taxonomy using other parts of the concept tree as needed.

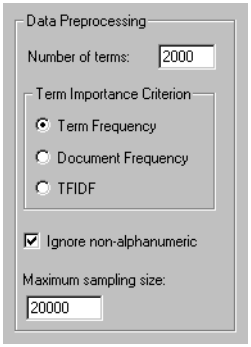
Only the categories associated with topics will give results. You will get no results if you test a category on a category from a concept node that has only term nodes.

Options and Parameters

This section describes the options and parameters listed under the Thematic Mapping Property page.

Data Processing

The Data Processing dialog box contains setting for number of terms and term importance criteria.



Number of Terms

The number of key terms (based on the Term Importance Criteria) you want to process and consider for inclusion in the concept tree.

Term Importance Criteria

The criteria for measuring the importance of a term.

- Term Frequency—the total number of times a term occurs in the document set.
- Document Frequency—the number of documents in the set in which the term occurs.
- TFIDF—Term Frequency defined above, divided by the Document Frequency defined above.

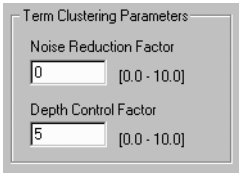
Additional Data Processing Options

Additional options are provided to:

- Ignore nonalphanumeric data—checking this option confines the data to alphanumeric characters only.
- Set a maximum sample size—sets the maximum size for the sample documents. If the input has more documents than the number specified in this option, a linear sampling is performed to limit the number of documents to this maximum number. If the input has fewer documents than the number specified, all input documents are used. The default value of this number is 20,000. You can increase this number if you have enough memory on your machine.

Term Clustering Parameters

The Term Clustering Parameters dialog box contains setting for noise reduction and depth control.



Noise Reduction Factor

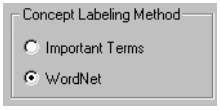
Controls the degree of noise filtering performed during the processing of statistical data collected from the input documents. See "Advanced features" for details.

Depth Control Factor

Controls the depth of the hierarchy generated. Increasing the value of this parameter will increase the depth of the hierarchy generated, and vice versa.

Concept Labeling Method

The Concept Labeling Method dialog box contains setting for important terms and WordNet.



Important Terms

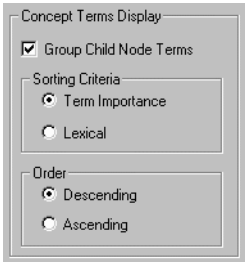
When selected, a concept is labeled with the two most important terms of the concept, according to the Term Importance Criteria selected.

WordNet

When selected, a concept is labeled with two terms selected that are most descriptive of the concept as a whole. These two terms are derived from the overall content of the concept using a proprietary algorithm with the WordNet Lexical Database serving as a knowledge source. (WordNet is not available in all locales.)

Concept Terms Display

The Concept Terms Display dialog box contains setting for grouping child node terms, sorting criteria, and order.



Group Child Node Terms

When selected, terms under different child nodes are displayed in groups represented by different colors; if deselected, all terms under the current node are displayed in a single group.

Sorting Criteria

The criteria by which the terms are sorted.

Order

The order in which the terms are sorted.

Sorting Child Nodes

The child term nodes and child concept nodes under each concept node in the tree are sorted in alphabetical order after tree creation.

Configuring Stopwords for Concept Extraction and Concept Labeling

To ensure that particular terms do not appear in concepts, you can add these terms to `vdk30.stp` under the corresponding locale directory. You might need to re-start Intelligent Classifier after changing `vdk30.stp` in order to make the changes effective.

WARNING! The change in `vdk30.stp` might affect VDK indexing and LRC topic creation.

There are two files (`labels.stp` and `terms.stp`) in Intelligent Classifier's installation directory under `_nti40\concept\labeling\stopword`. If you want to avoid the use of certain terms in deriving concept names, you can add these terms in `terms.stp`.

Thematic Mapping from the Command Line

From the command line, you can use `mktm` to automatically extract key concepts contained in a set of documents and organize them into a hierarchy—a concept tree.

To perform thematic mapping on a collection, run `mktm` and specify the collection to be processed with `-collection`. You can output the generated concept tree as an `.xml` file that complies with the verity taxonomy definition (taxonomy.dtd) or as a taxonomy in parametric index. To output the generated concept tree as an `.xml` file, specify the output file with `-exportct`. To output the generated concept tree as a taxonomy as a parametric index, specify both the parametric index path and the taxonomy name with `-pipath` and `-tax` respectively. If the parametric index specified does not already exist, it is created. For example:

```
mktm -collection c:\sample_collectoion -exportct c:\out\
sample_ct.xml -pipath c:\pi\my_pi -tax sample_ct_tax
```

To generate topics from the concepts in the generated concept tree, also specify, with `-topicset`, the location of the topic set in which to place the newly generated topics. If the topic set specified does not already exist, it is created. For example:

```
mktm -collection c:\sample_collectoion -exportct c:\out\
sample_ct.xml -topicset c:\ts\my_topicset
```

To generate a concept tree from a subset of documents in a collection, use the `-query`, `-parser`, and `-threshold` options. Only the documents retrieved by the query whose score is above the threshold are used for concept extraction.

```
mktm -collection c:\sample_collection -exportct c:\out\
samplect.xml -query "Verity Intelligent Classifier" -threshold 0.5
```

Alternately, you can use `-bif` to specify the documents to be used in concept extraction.

You can also generate topics from concepts in an existing concept tree by first importing it from a XML concept tree file. To do this, run `mktm` and specify the XML concept tree file to be imported with `-importct`, and specify the topic set to be used with `-topicset`.

```
mktm -importct c:\out\sample_ct.xml -topicset c:\ts\my_topicset2
```

`mktm` caches intermediate data generated in the thematic mapping process so that you can alter a number of parameters to effect the generated concept tree without `mktm` re-doing the data preparation process. These parameters include the noise reduction factor, depth control factor and the concept labeling method. The cached data is stored in the directory specified with `-workpath`. The default is the current working directory.

mktm Syntax

The command-line syntax for the `mktm` command line tool is shown below.

```
mktm [<option>...]
```

Syntax Descriptions

Table 5-1 describes the `mktm` elements.

Table 5-1 mktm Syntax

Element	Description
`-workpath` *`workpath_dir`*	The `-workpath` argument specifies the name of the directory where concept tree and other files will be created or updated. Required.
`-importct` *`conceptTreeName`*	The `-importct` argument specifies the name of the concept tree to import.
`-collection` *`collection`*	The `-collection` argument specifies the name of the collection that contains the source documents.
`-locale` *`locale`*	The `-locale` argument specifies the locale that will be used. Default is uni.
`-charmap` *`charmap`*	The `-charmap` argument specifies the *character map* that will be used. Default is the default charmap of the specified locale by `-locale locale_name`. Always use the default in version 5.0.
`-bif` *`bif_file`*	The `-bif` argument specifies the bif containing the list of documents to be used from the collection. When this argument is not given, `mktm` will use all documents in the collection.

Table 5-1 mktm Syntax (continued)

Element	Description
`-query` *query_string*	The `-query` argument specifies the query to be used for extracting documents to be used in concept extraction. This argument cannot be used with `-bif`.
`-parser` *parser_name*	The `-parser` argument specifies the parser to pass the *query_string* from the -query argument. It must be one of the following: Simple, Boolplus, FreeText, or Prefix. Default is Simple.
`-threshold` *number*	The -threshold argument specifies the threshold used to retrieve documents with the query specified by -query. The value divided by 1000 is used as the threshold. Only those documents whose score is above the threshold will be used. Range is 0 to 9999. Default value is 5000.
`-maxterm` *max_num_terms*	The `-maxNumTerm` argument specifies the maximum of terms to process and consider for inclusion in the concept tree.
`-select` *Selection_Criteria*	The `-select` argument specifies the term selection criteria. It can be one of the following values: `0 -` Term Frequency: the total number of times a term occurs in the document set `1 -` Document Frequency: the number of documents in the set in which the term occurs `2 -` TFIDF: Term Frequency divided by the Document Frequency
`-samplesize` *number*	The `-samplesize` argument specifies the maximum sample size used in thematic mapping. If the input to the thematic mapping engine has more documents than the number specified by `samplesize`, a linear sampling is performed to limit the number of documents. If the input has fewer documents than the number specified, all input documents are used. Default value is 20,000. Set a value of -1 to use all input documents.
`-noise` *noise_reduction_factor*	The `-noise` argument specifies the noise reduction factor. Controls the degree of noise filtering performed during the processing of statistical data collected from the input documents. Can be between 0 and 1000. Default is 0.
`-depth` *depth_control_factor*	The `-depth` argument specifies the depth of the hierarchy generated. Increasing the value of this parameter will increase the depth of the hierarchy generated, and vice versa. Can be between -500 and 500. Default is 0.

Table 5-1 mktm Syntax (continued)

Element	Description
`-label` *`concept_label_method`*	The `-label` argument specifies the concept labelling method. It can be one of the following values: `0` - Important Terms: a concept is labeled with the two most important terms of the concept, according to the Term Importance Criteria selected. `1` - Word Net: a concept is labeled with two terms selected that are most descriptive of the concept as a whole. These two terms are derived from the overall content of the concept using a proprietary algorithm with the WordNet Lexical Database serving as a knowledge source. (WordNet is not available in all locales.)
`-pipath` *`pi_directory`*	The `-pipath` argument specifies the path to the parametric index directory.
`-tax` *`taxonomy_name`*	The `-tax` argument specifies the name of the taxonomy to output the concept tree
`-exportct` *`export_concept_tree`*	The `-exportct` argument specifies the name of the concept tree to be generated.
`-topicset` *`topicset_directory`*	The `-topicset` argument specifies the path to the topic set.
`-vdkhome` *`vdkhome_directory`*	The `-vdkhome` argument specifies the Verity common directory.
`-tmhome` *`tmHome_directory`*	The `-tmhome` argument specifies the thematic mapping home directory
`-verbose`	The `-verbose` argument instructs `mktm` to run in verbose mode.

Troubleshooting

This section provides solutions to some of the more common issues.

The generated concept tree is too flat or there are too many top level concepts

Adjust the Depth Control Factor in the property page. Increasing the value of this parameter will increase the depth of the hierarchy generated, and vice versa.

The generated concept tree is incoherent or is very small

The problem usually occurs because of a lack of data, which is often due to the use of a small number of input documents or the lack of content in the documents. Other than the amount of input data, the quality of the resulting concept tree also depends on the quality of the input documents. For example, web pages containing a large number of templates, and long documents containing segments of unrelated content will not be able to provide good input data for the thematic mapping process.

When the concept tree lacks data, try the following:

- Increase the number of documents used for generating the concept tree.
- Increase the Noise Reduction Factor in the property page (the value should be increased gradually, using a value that is too large can lead to loss of data and result in deterioration of the quality of the concept tree).

When the quality of the input documents is low, try one or both of the following:

- Preprocess the documents to remove contents that are not related to central theme of the documents.
- When the input documents are very large, create several smaller documents out of the larger ones.

The thematic mapping process is slow

The thematic mapping process uses a large amount of memory. The process can be very slow when there is not enough physical memory available and virtual memory is used instead. If the process is very slow, try to:

- Increase the amount of physical memory available.
- Decrease the number of terms to process in the property page. (See "Setting the Concept Extraction Criteria" on page 106.)

When I try to open a saved concept tree I get this error: "the term file is missing"

Ensure the matching `.trm` file of the `.ct` file you are trying open is located in the same directory as the `.ct` file. (See "Concept Tree Management" on page 108.)

Altering the value of the "noise control" parameter does affect the depth of the tree. But change in the depth is only one of the many effects that will result from altering the parameter. And it is not advisable to alter this parameter when the goal is only to change the depth of the generated tree.

We have run tests with different "properties" settings and are curious about "Noise control" versus "Depth control"

The *noise control* parameter controls the amount of *noise filtering* performed. More noise filtering (a higher value for this parameter) means that the system will only treat concepts derived with high degree of confidence as valid. The result is usually a smaller (both in depth and in width) tree but one of higher quality. The default value of the parameter is 0 (minimum noise filtering).

Usage of the Noise Control Parameter

When a small or noisy document set is used as input, and the quality of the generated concept tree is found to be less than satisfactory, then increasing the value of this parameter may improve the quality of the tree.

Note When the input document set is neither small nor noisy, then increasing the value of this parameter will only result in the exclusion of useful data and the generated tree will contain fewer terms and concepts. Using too high a value for this parameter may also lead to extensive loss of data and result in deterioration of the quality of the generated tree. You are advised to adjust this parameter gradually and observe the changes in the quality of the result with respect to the changes to determine the optimal setting.

The *depth control* parameter is for controlling the depth of the tree. The concept tree is constructed bottom up: starting with the terms, and then the base level concepts, the tree grows upwards. When this "growing" stops, all the resulting subtrees are grouped under the root node, forming the final tree. The depth control parameter controls how far up grow the tree. Changing this parameter **may** only affect the depth of the tree, but not the numbers of terms or base concepts contained in it (unlike the noise control parameter).

Using of the Depth Control Parameter

For a deeper tree containing fewer top level subtrees, increase the value of the parameter; conversely, for a shallower tree containing more top level subtrees, decrease the value.

Note Altering the value of this parameter **may**, but not always, result in the change of the depth of the tree. The depth of the generated tree depends on both the semantic closeness between the concepts/subtrees in the tree, and the value of this parameter. For example, when the generated concept tree contains ten totally unrelated (that is, semantically very distant) top level concepts/subtrees, increasing the depth value by any amount should not and will not force these totally unrelated concepts/subtrees to be grouped together (thus, increasing the value of the parameter will not affect the depth). Similarly, when concepts under each of the top level subtrees are very closely related, decreasing the value of the parameter may not affect the depth of the tree.

Thematic Mapping with default PDF filter (flt_kv)

Thematic Mapping is not supported for PDF documents with the PDF filter (flt_pdf) because this filter does not extract noun and noun phrase information from PDF documents. Use the Keyview filter (flt_kv) to index PDF documents.

For more information about the `rcadmin` command-line tool, see the *Verity K2 rcadmin Guide*.

PART II

Defining Categories and Creating Topics

Categories organize documents into subjects of interest with category-to-document relationships stored in a category index database in the knowledge tree.

One important element of a category is the classification rule that determines what documents will be associated with the category. Classification rules are defined through category properties, which include:

- Rule
- Threshold
- Refining Category

A rule can be a simple query (VQL) expression, a topic in a topic set, or a complex query with mixed topics and simple query terms.

Topics are commonly used for defining classification rules of categories in a taxonomy. They are also heavily used in other Verity applications, such as Verity Profiler and search applications.

This part focuses on various ways for building topics. It contains the following information:

- Using the Topic Editor
- Creating Topics Using Logistic Regression Classifier
- Creating Topics Using Hybrid Approaches

For information about designing and building topics, see the *Verity Query Language and Topic Guide*.

6

Using the Topic Editor

Topic sets can be stored in either binary or OTL format. Intelligent Classifier's graphical topic editor supports both formats. You can open a topic set in one format in the topic editor and export it to the other format.

Note You must use the OTL format for topic set encryption and indexing. To encrypt topics in a topic set or to index topics into a collection, use the `mktopics` command-line tool. For information about how to use `mktopics`, see the *Verity Query Language and Topic Guide*.

The topic editor provides various just-in-time assists to facilitate your editing process. It also validates newly entered topics.

You might need to load collections if your topics involve collection specific information, such as fields or zones.

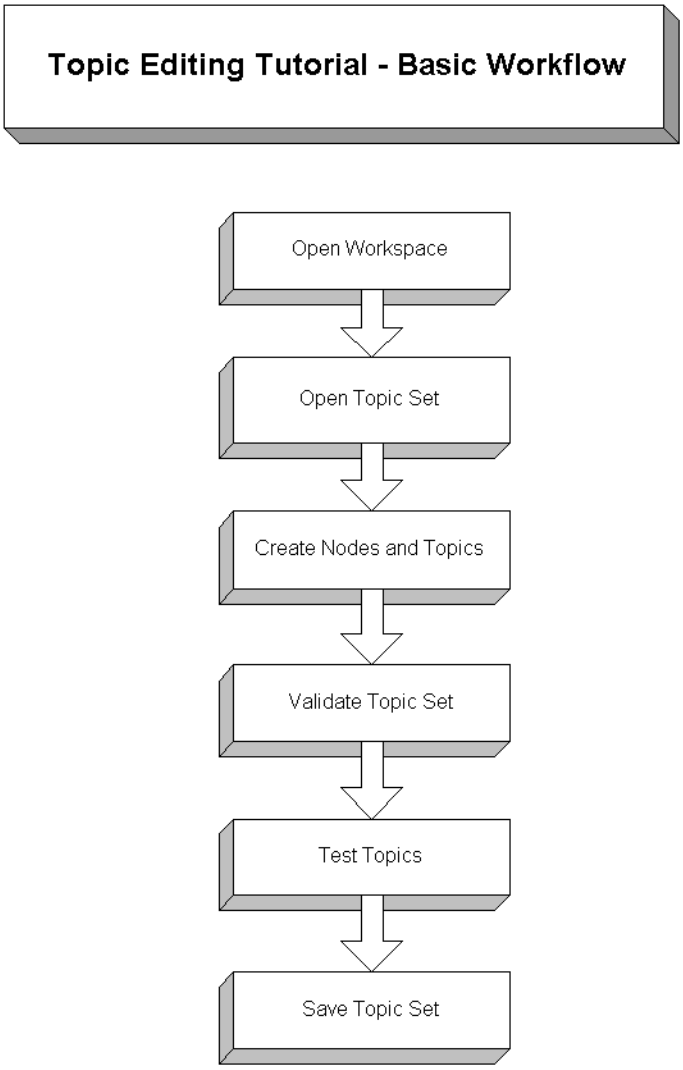
This chapter discusses how to edit a topic set using Intelligent Classifier.

Information in this chapter includes:

Topic Editing Tutorial

This section demonstrates how to create and test topics. (Topics are predefined queries that end users can use.) Figure 6-1 shows the basic workflow for this tutorial.

Figure 6-1 Topic Editing Tutorial—Basic Workflow



Tutorial Basics

First, you will create a new Intelligent Classifier workspace.

1. Launch Intelligent Classifier.

   The first screen you see depends on the Startup settings in the Options dialog box. You can set Intelligent Classifier to do any of the following at startup:

   - Immediately enter a Wizard where you can create a new workspace and automatically generate a new topic set and taxonomy.
   - Automatically open your most recently used workspace.
   - Automatically open your most recently used Knowledge Tree.
   - Open Intelligent Classifier with no files loaded. You can then create your workspace, topic set, and taxonomy separately.

   In this tutorial you create a workspace and topic set separately. However, you can also create a workspace and topic set at the same time using the Classification Wizard.

2. If the **Classification Wizard** window is already displayed, close it.

3. Select **File | New Workspace**.

   The New Workspace dialog appears.

4. In the **Name** field, enter `tutorial` for the name of the workspace.

5. Enter a location for storing the Intelligent Classifier files in the **Location** field. You can either type a location or find a location and insert the path where the Intelligent Classifier files will be located.

   For this tutorial save the workspace using this path:

   `IC_install``\Verity\IC\samples`

   Where *`IC_install`* is the directory path where Intelligent Classifier has been installed.

   Use the button containing ellipses (. . .) to the right of this field to browse and find a location.

6. Click **OK**.

Creating Nodes and Topics

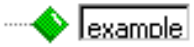
This section demonstrates how to create nodes and topics. (Nodes are elements in the topic tree. A node can be an unnamed node, or a named node. Named nodes are topics.)

Add Top Level Nodes

1. Choose **Edit | Add Top level Node**, or click the **Node** icon

You have just created an empty node.

2. Type `example` as the name for this node. (Top level nodes always need to be named, but lower-level nodes can be named or unnamed.)

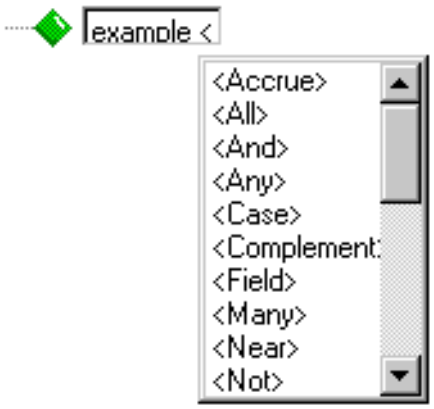


While you are in *editing mode*, Intelligent Classifier draws the node in a box. This is similar to the way Windows Explorer indicates that the name of a file or directory is being edited.

Note Topic names cannot include spaces.

3. Type a left angle bracket (<).

The operator assist list box appears. This box shows all the operators that are valid at this point.

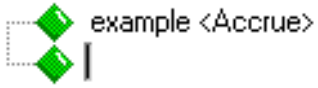


Either:

- Type the complete operator yourself (including the opening and closing angle brackets).
- Select the appropriate operator from the list by clicking on it.
- Start typing the name of an operator.

  If you start typing a name, Intelligent Classifier will scroll to the first operator that matches what you have typed. As soon as the operator you want is highlighted in the list, press the **Spacebar** to accept it.

For this example, select the `<Accrue>` operator and press the **Spacebar** to accept it. Then press **Return** to finish editing the node.



Note Values with spaces must be surrounded by quotes. In the following example, Mark Twain must be surrounded by quotes:

```
example <Field> Author <Contains> 'Mark Twain'
```

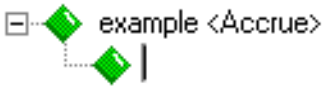
Adding Non-Top-Level Nodes

1. Intelligent Classifier automatically starts another sibling node, but in this case you want to add more children to this topic. Press **Return** or **Esc** to dismiss the newly-created node.

Pressing the **Esc** key cancels any edits you are making to an existing node, and erases any newly-formed, empty node.

2. Select the `example` node (by clicking on it), and Choose either **Edit | Add Child Node**, or click the **Child** icon.

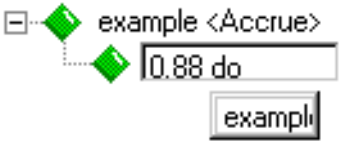
A new child node appears.



3. Give this node a weight and a name.

   a. Type in weight of `0.88`. (*Weights* affect how the nodes score the documents.)

   b. Type `document_viewing` for the node's name. The Topic Assist appears.



The assist lists all the names of topics in this topic tree. To re-use an existing topic, select it from this list. (See "Moving and Sharing Nodes" for an example.)

4. In this case, keep typing `document_viewing` and choose `<Phrase>` for the operator.

   If you try to type in a search item after the `<Phrase>` operator, Intelligent Classifier will not let you because the `<Phrase>` operator does not accept any search terms. This is an example of how Intelligent Classifier helps you avoid creating any syntactically invalid topic sets.

5. With the node still selected and still editable, press **Shift-Return**. This is a shortcut to finish editing this node and start editing a new child node.

6. Do not give this node a name or weight, so:

   a. Type `<` to start entering the operator, and select `<Word>`.

   b. This operator does need a search term, so type `document`.

   c. Press **Return** to finish editing this node and create another child node beneath `document_viewing`.

7. For the next node, enter `<Stem> view` and press **Return**.

If both child nodes of the `<Phrase>` operator had used the same operator (that is, both had used `<Word>` or both had used `<Stem>`), you could have used an editing shortcut.

Two of the nodes (`example` and `document_viewing`) are shown in blue, although the others are shown in black. This indicates that `example` and `document_viewing` are *topics*.

Adding a Sibling Node

1. Select the `document_viewing` node and choose **Edit | Add Sibling Node**, or click the **Sibling** icon.

   An empty node appears.

2. Simply enter `<Or>` for this node.

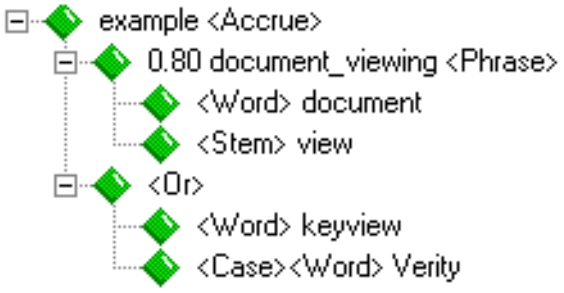
3. Underneath that node, create a child node containing `<Word> keyview`.

   Now create a sibling node to the `<Word> keyview` node. Enter a modifier as well as an operator.

4. Create the node. Type `<` and selecting `<Case>`. (This is a modifier that sets the `<Word>` operator to be case-sensitive.) Now type `<` again. The drop-down list now contains only the operators and modifiers that are consistent with what has already been entered. This is another example of how Intelligent Classifier helps you create syntactically valid queries.

5. Choose `<Word>`, then type `Verity` to complete the node.

   At this stage the topic is complete. It should look like this:



Moving and Sharing Nodes

This section of the tutorial describes how to move and share nodes. Shared nodes are nodes that are reused in more than one place in the topic tree. Changes in one place affects all the other shared copies.

Open the Outline File and Expand the Tree

First, open the outline file named `tutorial_share.otl`:

1. Choose **File | Open Workspace**.
2. In the Files of Type popup, select **Topic Set Files (.otl)**.
3. Select `tutorial_share.otl`, located in the samples directory.

   The sample files are located in *`IC_install`*`\samples\` (where *`IC_install`* is the directory where Intelligent Classifier is installed, typically `C:\Program Files\verity\intelligent classifier`).

   Intelligent Classifier opens the topic set and creates a workspace for it.

Expand the topic tree so you can see the lower-level nodes:

1. Select `products_1` and Choose either **Edit | Expand** or click the **Expand** icon.

2. Select `products_2` and Choose either **Edit | Expand** or click the **Expand** icon.

   Notice that the `<Case><Word> Server` nodes have double diamond icons. This indicates that they are shared.

Dragging and Dropping Nodes

1. Select the `topic_creator` node and drag it onto the `products_2` node.

   When you release the mouse, the node is moved to the new location. (Intelligent Classifier warns you if you try to drag it onto a node that does not accept children.)

2. **Ctrl-drag** (hold down the Ctrl key *after* you start to drag) the `<Phrase>` node for "Verity Spider" onto `products_1`.

   This example demonstrates that you can drag and drop *all* nodes, including unnamed ones (not just named nodes). It also shows how Ctrl-dragging creates a *shared copy* of the node instead of actually moving the node.

Because this node is shared, changes made to it in one place are reflected in all shared copies.

Note You can also drag and drop, or copy a topic to the taxonomy pane to generate a new category. If you create a topic with named children and copy the topic to the taxonomy pane, this first copy of the topic (its first appearance in the taxonomy pane) will show the topic's children. If you copy the same topic from the topic pane to somewhere else in the taxonomy pane, this second copy references the first copy, and will not show any children.

Modifying Nodes

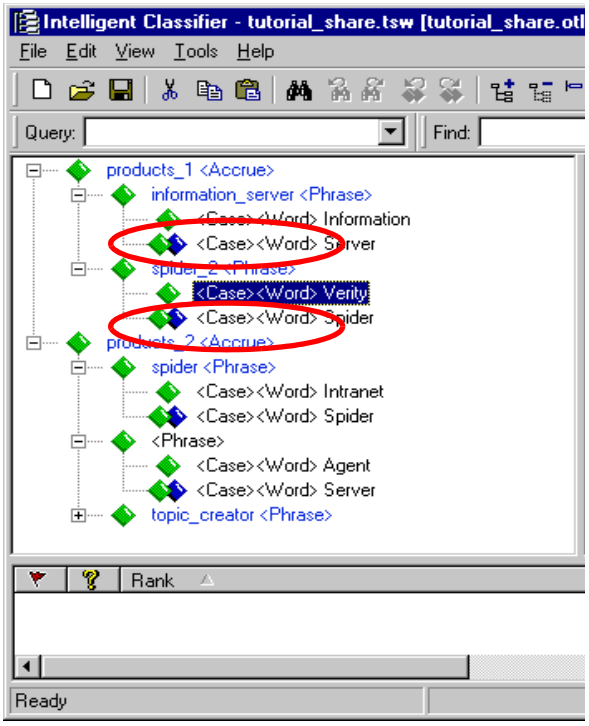
1. In the child of the first shared `<Phrase>` node, change the word `Verity` to `Intranet`. (Select the node, move the cursor to the end, backspace over `Verity`, and type in `Intranet`.)
2. Name the `<Phrase>` node `spider`. (Select the node, move the cursor to the beginning of the node, and type the name in.)

   Notice that the changes affect both instances of the shared node.

Unsharing Nodes

1. Select the first `spider` node and choose **Edit | Break Link**. This stops the sharing, and makes the selected node into an independent node.

   Notice that Intelligent Classifier has renamed the first node to `spider_2`, and that the node's icon shows it is no longer shared. Its child nodes still have the double-diamond icons because they are still shared.
2. Click on `spider_2`'s first child node and choose **Edit | Break Link.**
3. Change the word `Intranet` to `Verity`.



This change only affects the edited node because this node is no longer shared.

4. Create a top level node.

   `products_3 <Accrue>`

5. Create a child node under it.

   Type the letter `t` in the empty child node.

   Notice that as soon as you start typing, Intelligent Classifier displays an assist with the names of all the existing topics. Press the **Spacebar** to accept `topic_creator` and press **Return**.

Note You do not have to type in the operator, just select the name of the topic. Intelligent Classifier inserts a shared copy of that topic.

Using Drag and Drop Features

This section demonstrates how to create nodes using drag-and-drop features.

Selecting a Topic Set and Document

Open `tutorial_drag.otl`:

1. Choose **File | Open Workspace**.
2. In the Files of Type popup, select **Topic Set Files (.otl)**.
3. Select `tutorial_drag.otl`, located in the samples directory.

   The sample files are located in *`IC_install`*`\samples\` (where *`IC_install`* is the directory where Intelligent Classifier is installed, typically `C:\Program Files\verity\intelligent classifier`).

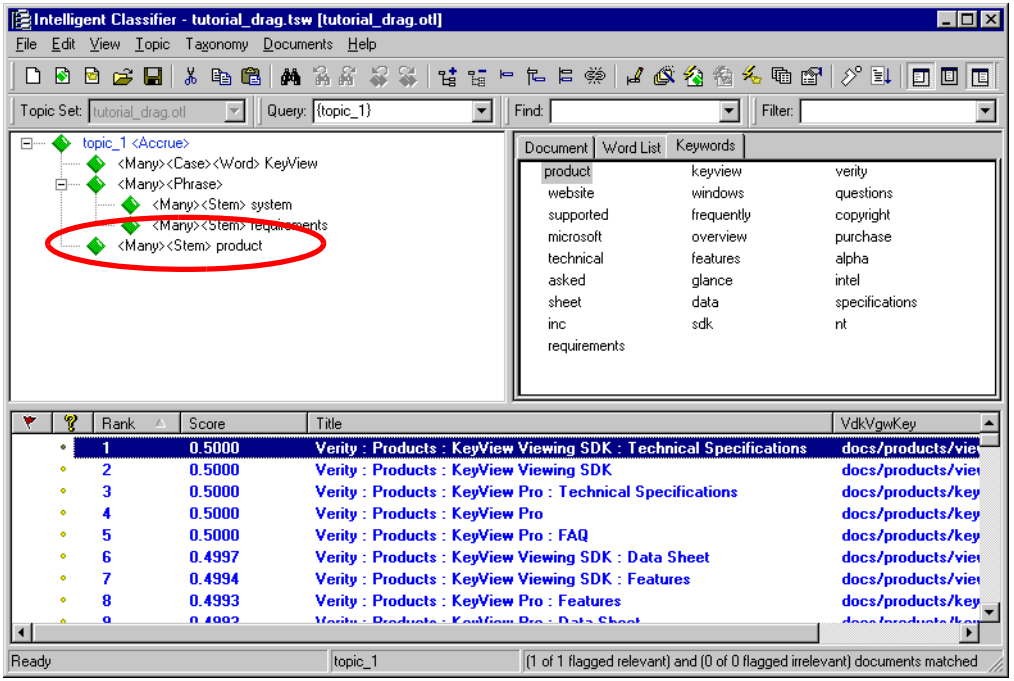
   Intelligent Classifier opens the topic set and creates a workspace for it.
4. Select **File | Add/Remove Collections**. The `vrtyweb` collection should appear in the "Recently Used" pane.
5. Select on the `vrtyweb` collection.
6. Click **Add**.
7. Click **OK** to close the dialog.
8. Select `topic_1` and test it.
9. In the Search Results pane, select the first document.

Drag and Drop Actions

1. In the Document View pane, select the phrase `system requirements` and drag it onto `topic_1`.

   Intelligent Classifier creates a new node with the selected phrase.
2. In the Document View pane, click the **Keywords** tab. This shows the most important words in the selected document.
3. Drag the word `product` onto `topic_1`.

   Intelligent Classifier creates a new node with the selected word.



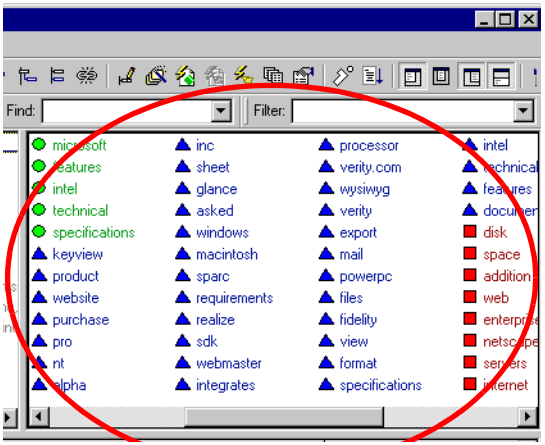
4. Drag `specifications` from the **Word List** tab onto `topic_1`.

   These examples show how Intelligent Classifier makes it easy to create new evidence nodes.

You can also drag-and-drop words from other drag-and-drop aware Windows applications onto Intelligent Classifier.

You can generate clusters of words without creating new topics. With the first ten documents selected, choose **Documents | Extract Document Clusters**.

Intelligent Classifier shows the clusters in the Document View pane. The clusters are indicated by different colors and icons.



The number and depth of clusters can be controlled though the property settings. You can drag and drop words from the cluster view onto a topic.

Validating a Topic Set

1. Choose **Topic | Validate Topic Set**. This checks that all operators that require child nodes under them have been filled in with child nodes. For this topic set, you should see no errors.
2. Continue with "Testing a Topic."

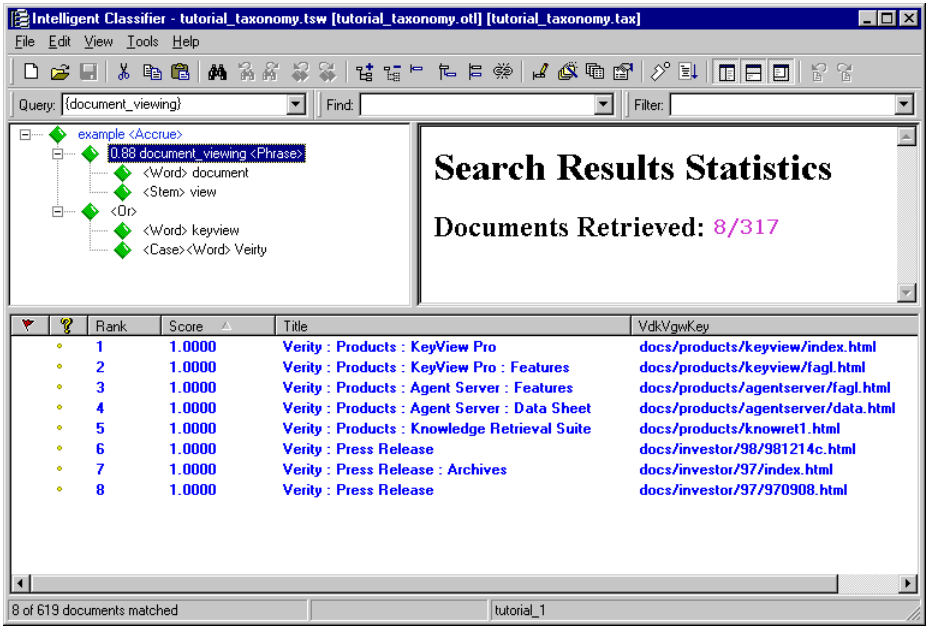
Testing a Topic

This section demonstrates how to *test* a topic. That is, how to use the topic to search the documents in the collection.

In the topic tree, select the `<Or>` node. Notice that **Topic | Test Topic** and the **Test** icon are unavailable. This is because only named nodes (topics) can be tested.

1. Select the `document_viewing` topic. **Topic | Test Topic** and the **Test** icon are now available.
2. Choose **Topic | Test Topic** (or click the toolbar icon in the toolbar, or double-click the topic in the topic tree, or type **Ctrl+T**).

The Document View pane lists the search result statistics and the Search Results pane lists documents found by the search. The maximum number of documents listed in the Search Results pane can be specified though the properties.



The Statistics line at the bottom of the window shows how many documents were retrieved out of the total number in the added collection(s).

All the documents listed in the Search Results pane are shown in blue boldface. This indicates that they have not been viewed and have not been retrieved before within this workspace.

3. Double-click any document in the Search Results pane.

   The Document View pane shows the document.

4. Now select another document in the Search Results pane.

   Notice that the first document you viewed is no longer shown in boldface in the Search Results pane. This indicates that you have already viewed this document.

5. Test the topic again. Notice that now the documents are listed in black in the Search Results pane. This indicates that these have been retrieved before. This provides an easy way to see, as you edit a topic, what new documents are being retrieved.

Saving Topic Sets

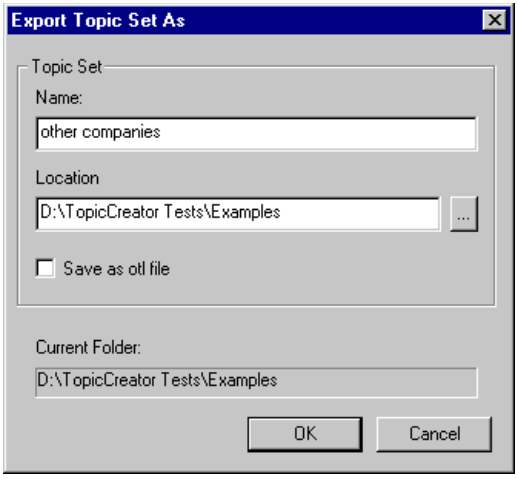
You can use either **File | Save All** or **Topic | Save Current Topic Set** to save the topic set. You can also use **Topic | Export Topic Set** to save the topic set to a different location in either binary or OTL format. The following sections describe how to save your topic sets in different formats.

Saving Topic Sets in Original Format

To save the current topic set, choose **Topic | Save Current Topic Set** (or press **Ctrl-S**, or click the **Save** icon).

Exporting Topic Sets

1. To create a copy of the topic set that you can export as an ASCII .otl file, choose **Topic | Export Topic Set**. The Export Topic Set As dialog box appears.

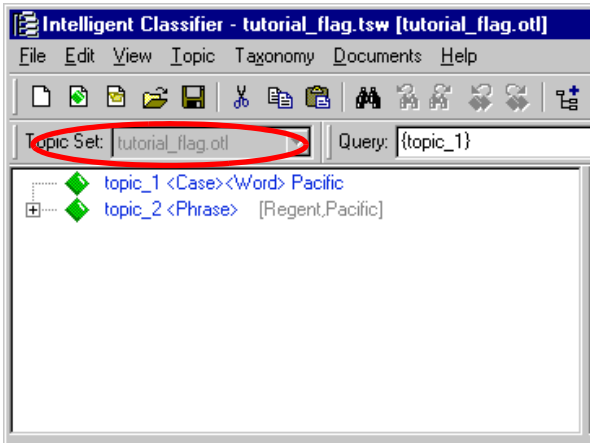


2. Enter the name and location for the topic set.
3. To save the topic set as an OTL file, check **Save as otl file**.
4. Click **OK**.

Intelligent Classifier saves a copy of the topic set but continues using the topic set with the current name.

Getting Information About Topics

The following subsections describe how to get information about your topics.



Most of these documents contain the phrase “Regent Pacific” but two do not. Six documents are flagged as “relevant” and two as “irrelevant”. (These flags are visual markings to help track which documents are being retrieved. They do not affect the search results in any way.)

Choose **File | Properties** to view the Properties dialog.

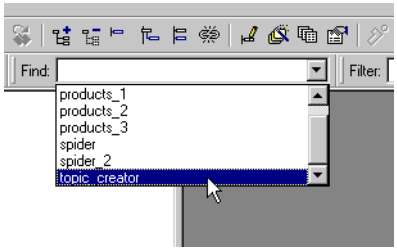
Finding Topics

To find a topic in the topic tree, select it in the Find Topic list. (See “Intelligent Classifier User Interface.”)

Intelligent Classifier populates this list with the names of all the topics in the active topic set. To have Intelligent Classifier scroll to the appropriate section of the list, type the first letter of a topic’s name.

Finding Instances of Topics

1. Using the Find list, select `topic_creator`.



Intelligent Classifier finds and highlights that topic.

2. Choose **Edit | Previous Use Of Node** or **Edit | Next Use Of Node** to find the other instance of `topic_creator`.
3. Choose **File | Save All**.
4. Choose **File | Close Workspace**.

Flagging Document Relevance

This section discusses flagging documents. Flagging a document involves marking a document as relevant or irrelevant. This flagging provides visual clues about whether the documents being retrieved are relevant.

Retrieving Information

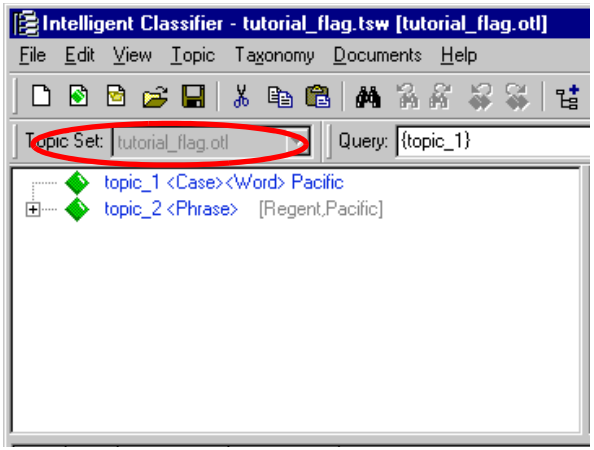
First, you will open `tutorial_flag.otl`:

1. Choose **File | Open Workspace**.
2. In the Files of Type pop-up, select **Topic Set Files (.otl)**.
3. Select `tutorial_flag.otl`, located in the samples directory.

   The sample files are located in *`IC_install`*`\samples\` (where *`IC_install`* is the directory where Intelligent Classifier is installed, typically `C:\Program Files\verity\intelligent classifier`).

   Intelligent Classifier opens the topic set and creates a workspace for it.
4. Add the `vrtyweb` sample collection.

   To search for documents that contain the phrase `Regent Pacific`, a first attempt at creating a topic might be `topic_1`.
5. Select topic `topic_1` and test it.



Most of these documents contain the phrase `Regent Pacific`, but two do not. Six documents are flagged as `relevant` and two as `irrelevant`. (These flags are visual markings to help track which documents are being retrieved. They do not affect the search results in any way.)

Flagging Relevance Settings

Tell Intelligent Classifier to use one set of relevance settings for all the topics in the workspace, rather than using a separate set for each topic.

1. Choose **File | Properties**. The Properties dialog appears.
2. On the **General** tab, select **All topics use the same relevance settings**, and click **OK**.
3. Mark `970214b.html` as irrelevant. In the Search Results pane, in the column underneath the flag icon, click beside the document.

   The first time you click, a green checkmark appears. If you click again, the checkmark turns into a red cross. If you click again, the cross disappears and you return to the blank state. Click until the red cross is shown beside this document.

4. Repeat with the document `970203b.html`.
5. Now mark all the other documents as relevant.
6. Test `topic_2`.

   The search results statistics area at the bottom of the window now tells us how many flagged documents this topic returned. Here it has returned all six of the documents flagged as relevant, and it has not returned either of the documents flagged as irrelevant.
7. If you test `topic_1` again, you will see that it returns seven documents flagged as relevant and two documents flagged as irrelevant.
8. Choose **File | Save All**.
9. Choose **File | Close Workspace**.

You can flag all the documents in the Search Results pane by choosing **Documents | Document States | Flag Displayed Documents as 'Relevant'** and **Flag Displayed Documents as 'Irrelevant'**. You can retrieve all flagged documents by choosing **Documents | Return Flagged Documents**.

Using the Assists Dialog

Intelligent Classifier can show you the results of various operators and search terms for the current collection(s) and can show you the fields and zones defined in the collection(s).

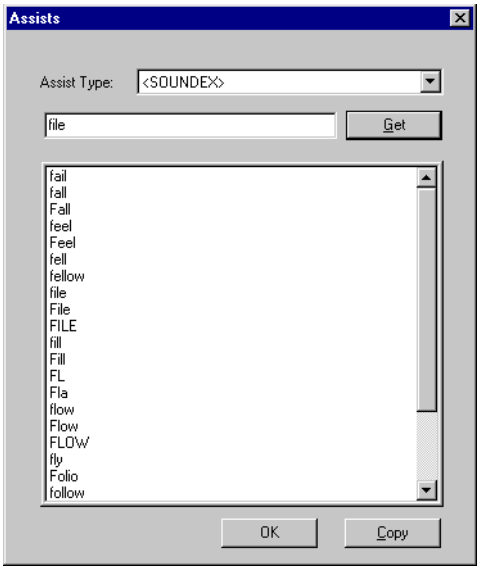
Note Zone assist is not available for the collection without a spanning word list.

To use these assists:

1. Choose either **Topic | Assists**, or click the **Assists** icon.

   The Assists dialog opens.



2. Choose the appropriate item in the **Assist Type** drop-down menu.
3. If you selected an operator, type a search term in the text field beside the **Get** button.
4. Click **Get**.

   If you selected an operator, Intelligent Classifier then displays all the words in the collection that match the given operator and search term. For example, selecting `<Soundex>` and `file` shows all the words that match a node designated:

   ```
   <Soundex> file
   ```

   If you selected `Fields` or `Zones`, Intelligent Classifier lists all the fields or zones for the open collection(s).
5. Click **OK** to close the Assists window.

You can select one or more words in the list and click **Copy** to copy them to the clipboard. You can then paste them onto the topic tree.

Choosing `<Wildcard>` and entering `w*`, for example, lists all the words in the collection starting with `w`. This provides an easy way to see what words the collection contains.

The Assists dialog only shows the `<Stem>`, `<Wildcard>`, and `<Word>` assists if the collection has *word list assist* enabled. It only shows the `<Typo/2>` assist if the collection has the *ngram index* added. The Assists dialog only shows the Zone assist if the collection has zones defined in it.

Explaining Topics

Intelligent Classifier's *explain* feature shows which parts of a topic are responsible for a particular document being selected, and shows how each of those parts contributes to the document's score. You can quickly see which parts of your topics are selecting different documents. It is also a useful tool to use when learning how different operators work.

Note Intelligent Classifier's *explain* feature works only when Intelligent Classifier has access to the original document and the topic set is not encrypted. The *explain* feature needs to profile the original document.

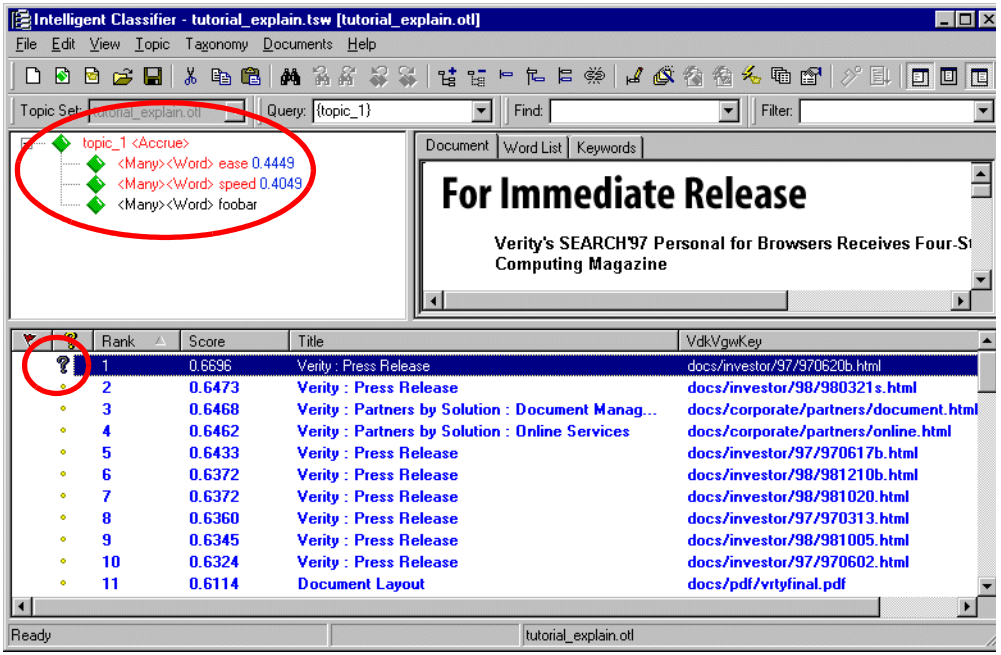
Using Explain

Open `tutorial_explain.otl`:

1. Choose **File | Open Workspace**.
2. In the Files of Type popup, select **Topic Set Files (.otl)**.
3. Select `tutorial_explain.otl`, located in the sample files.

   The sample files are located in *`IC_install`*`\samples\` (where *`IC_install`* is the directory where Intelligent Classifier is installed, typically `C:\Program Files\verity\intelligent classifier`).

   Intelligent Classifier opens the topic set and creates a workspace for it.
4. Add the `vrtyweb` sample collection.
5. Select `topic_1` and test it.
6. In the Search Results pane, select the first document.
7. In the column headed by the question mark icon, click beside that document, or right-click on the document and select **Explain** from the shortcut menu. A question mark appears beside the document and the Topic pane changes.



The Topic pane illustrates why that document was retrieved by the topic and why it is scored the way it is. Nodes marked in red (`speed` and `ease`) are those that match the document. Nodes marked in black (`foobar`) do not match the document. In this case, two of the topic's child nodes matched the document, and so the `<Accrue>` operator, which selects items that match at least one of the child nodes, matched the document and is also marked in red.

The numbers to the right of each node show the nodes' score. In this case, the `<Many><Word>ease` node returned a score of 0.4449 for this document. (The document's final score as shown in the Score column in the Search Results pane, 0.6696, is determined by how the `<Accrue>` operator combines the scores of the child nodes. See the *Verity Query Language and Topic Guide* for more information.

8. Choose **File | Save All**.
9. Choose **File | Close Workspace**.

Opening Encrypted Topic Sets

You can open an encrypted topic set in Intelligent Classifier. However, only the top-level topic nodes are displayed in the topic pane. No edits can be made to the topic set.

You can test top-level topics in the encrypted topic set and associate these topics with categories in a taxonomy.

For the complete syntax for `mktopics` and how to build encrypted topic sets see the *Verity Query Language and Topic Guide*.

7

Creating Topics Using Logistic Regression Classifier

This chapter describes Logistic Regression Classifier (LRC). It contains the following sections:

Introduction to Logistic Regression Classifier (LRC)

LRC is an enhanced state-of-the-art machine-learning algorithm that can automatically create a business rule or a Verity Topic from a set of positive and optionally negative exemplary documents. Positive documents refer to documents that are relevant to the topic (or category) of interest. Negative documents are the opposite, that is, documents that are irrelevant to the topic (or category) of interest. LRC automatically learns a classification rule in terms of a Verity Topic such that the positive exemplary documents can be maximally distinguished from the negative exemplary documents using this rule in the presence of negative exemplary documents. During the training process, LRC automatically identifies important positive and negative evidence terms from the exemplary documents and computes a numerical weight for each evidence term. The weight value is positive for positive evidence terms and negative for negative evidence terms. The absolute value of a weight indicates the importance of its corresponding evidence term to the topic or category. The larger this absolute value is, the more important the evidence term is to the topic or category.

LRC Topics

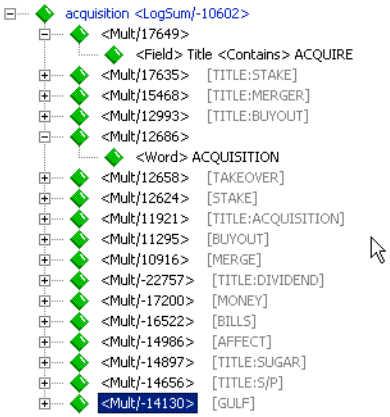
LRC Topics are Verity Topics created using LRC. Once an LRC Topic is created, it is inserted to the current topic set in Intelligent Classifier and displayed in the topic pane automatically. The figure below shows an example of an LRC Topic on “acquisition”. All LRC Topics use the `<LogSum/n>` operator (see “Valid Verity Query Language Operators and Modifiers”) in the top-level named node, where the integer n is also automatically learned by LRC and can be different for different topics. The `<LogSum/n>` operator can have up to 1024 child nodes (evidence nodes). Each child node consists of a weight-evidence pair.

Weight is specified as a signed integer “w” in the `<Mult/w>` operator. An evidence node can take one of the following two forms:

```
<Word> SomeTerm
```

or

```
<Field> SomeField <Contains> SomeTerm
```



When an LRC topic is evaluated against a document, all the positive and negative evidences that appear in the document are accumulated according to the LOGSUM formula. The score is always between 0.0 and 1.0. For more information about LOGSUM, see the *Verity Query Language and Topic Guide*.

Note It is possible to have a non-zero score for a document that does not contain any evidence, but normally this non-zero score is close to zero.

Creating a Topic or Updating a Topic

LRC can be operated in two modes:

- Creating a new topic
- Updating an existing topic

The first does not impose any constraint on the evidence terms, while the second confines its evidence terms to those already in the existing topic and recomputes weights for these evidences from a new set of positive and negative documents.

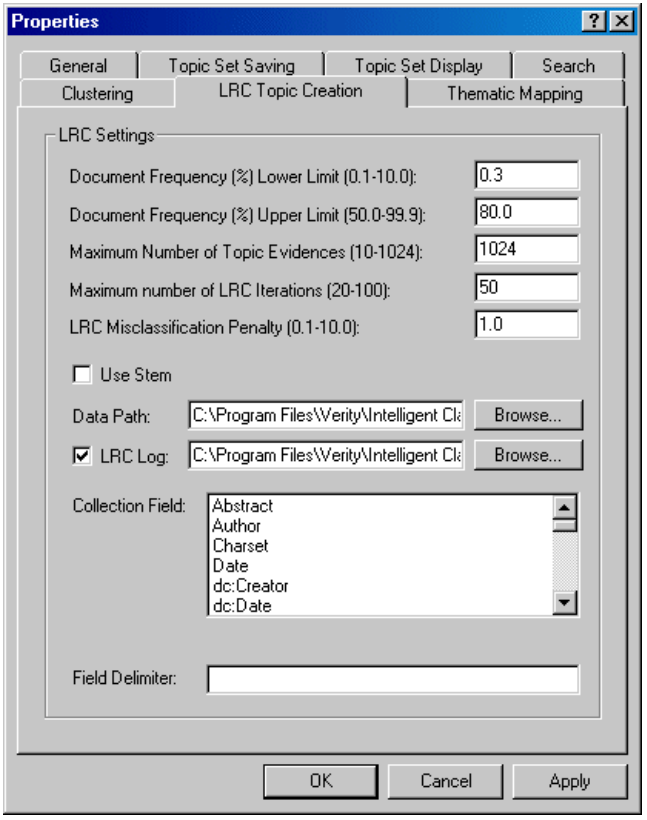
An existing topic can be a non-LRC topic, but the resulting topic will always be an LRC topic. When LRC generates a list of evidence terms from the existing topic, all the operators and weights are ignored.

When you invoke LRC to update an existing topic, ensure that the topic contains a sufficient number of evidence terms that appear in the training documents (positive and negative documents to the topic).

Options and Parameters

There are a number of user-configurable parameters for LRC. A user can go to the LRC Topic Creation properties page (**File | Properties**) to set these parameters as desired. The default parameter settings are shown in the figure below. These default settings should work reasonably well for most tasks. However, selecting non-default parameter values can lead to better accuracy and faster topic creation.

Note All these parameters are saved in Windows registry. When you launch the Intelligent Classifier application, the parameters used in the previous Intelligent Classifier session will be restored. Before you invoke LRC for topic creation, it is recommended that you check the parameter settings. When you run LRC with LRC logging enabled (see this option below), these settings will be logged into the LRC log file.



The following table shows the range, default value and description for each parameter.

Table 7-1 LRC Parameters

Parameter	Range	Default Value	Description
Document Frequency (%) Lower Limit	0.1-10.0	0.3	A term will not be considered if it appears in less than the specified percentage of the total number of documents in the collections. It is used to filter out terms that are too infrequent to be considered as important. A high value of this parameter means that fewer terms are considered as candidate evidence terms for LRC Topics.
Document Frequency (%) Upper Limit	50.0-99.9	80	This value divided by 100 will be used in LRC evidence selection. A term will not be considered if it appears in more than the specified percentage of the total number of documents in the collections. It is used to filter out common terms which cannot discriminate documents from different categories.
Maximum Number of Topic Evidences	10-1024	1024	The maximum number of evidences for every topic.
Maximum Number of LRC Iterations	20-100	50	LRC algorithm will end when this number of iterations is reached. LRC often finishes before that. A smaller value of this parameter means faster training for some topics, but may reduce accuracy.
LRC Misclassification Penalty	0.1-10.0	1.0	This is the weight on the mis-classification in the training process. See the important note below on this parameter.
Use Stem	Checked or Unchecked	Unchecked	When checked, word stems, rather than word tokens will be used as candidate features. When unchecked, word tokens will be used as candidate features.

Table 7-1 LRC Parameters (continued)

Parameter	Range	Default Value	Description
Data Path	Valid directory path	C:\Documents and Settings\userA\ Application Data\ Verity\Intelligent Classifier	LRC will use this path to save training data
LRC Log	Valid file path	C:\Documents and Settings\userA\ Application Data\ Verity\Intelligent Classifier\LRC\ lrc.log	LRC will append log information into this file. User can turn off logging by un-checking the check box.
Collection Field	A field in VDK collections		LRC will derive evidences from the selected field(s). Terms in the field(s) separated by a string specified in **Field delimiter** are used as evidence terms in addition to terms in the body text of documents.
Field delimiter	String		A delimiter for parsing the field data

Important Notes

Intelligent Classifier caches the training data for LRC under the **Data Path**. It creates one directory for the current set of collections. The directory name is a hash code from the collection paths. When LRC is invoked, it first checks if

- collections are the same as in the previous Intelligent Classifier session
- the training data have been prepared in the cache directory
- the cache is fresh (newer than collections)
- no parameter is changed in **Document Frequency Lower Threshold, Document Frequency Upper Threshold, Collection Field,** and **Field Delimiter**

If these conditions are met, LRC uses the existing cache. Otherwise, it recreates the cache.

If large collections are used, ensure that there is enough disk space on the drive that the **Data Path** is on.

- You can use **Document Frequency Lower Threshold** and **Document Frequency Upper Threshold** to filter out terms that are too frequent or too infrequent. A higher value for **Document Frequency Lower Threshold** and lower value for **Document Frequency Upper Threshold** mean that fewer terms will be considered as candidates for evidences in a topic.
- When large collections are used, it is recommended to use a value larger than the default for **Document Frequency Lower Threshold**, and a value smaller than the default for **Document Frequency Upper Threshold**. This will reduce the cache size because more terms will be filtered out.
- When a taxonomy is auto-generated from a metafield, this metafield should not be selected for LRC to derive evidence from. Otherwise, the resulting LRC Topics will not be able to accurately classify unseen documents that do not contain this metafield. Certain fields, such as **Title** and **Keywords**, contain important information about document content. It is strongly recommended to select these fields for creating LRC Topic creation whenever they are available in the collections.

  Note These metafields have to be indexed into the collection first.

- When LRC is invoked on auto-generated categories, categories with LRC topic rules are realized (they show up in the TAX file). However, the auto-parent node remains as an auto-parent category with one of the property options on the Auto Categorization page: (File, URL, Field) checked. The sub-taxonomy rooted at the auto-parent is still dynamic, that is, when docs in the collection changes, or new collections are attached, the categories under the auto-parent may change.

  To harden the taxonomy after LRC is called, right click the category and select **Properties**. Tab to the **Auto Categorization** page and select **None**.
- The default value of 1.0 for `LRC Misclassification Penalty` should work for typical topic creation. In general, the larger this value is, the more accurate the LRC Topics are on the training documents. However, it is not necessarily the case on other documents that are not in the training set. If the total number of training documents is small, it is not recommended to use a large penalty value for misclassification. A user is encouraged to test different values on test documents that are not used in the training set in order to select the best value for this parameter.
- If a large number of LRC Topics need to be created in a single topic set, it is recommended to use a small value for the **Maximum Number of Evidences** parameter. The larger the topics are, the slower is to test these topics and to generate a

knowledge tree using these topics. Moreover, some limits might be reached before all the topics are created. These limits include:

- The maximum topic set size which is 64 MB
- The maximum number of nodes in a topic set which is 5 million
- The maximum number of links in a topic set which is 8 million.

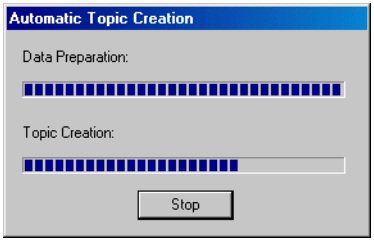
Creating and Updating Topics Using LRC

The collection needs to be optimized with a spanning word list. This can be done by letting Intelligent Classifier build the span word list when it finds the collection does not have a spanning word list for certain operations, like LRC topic creation and concept mapping. You can also do the optimization on the command line before starting Intelligent Classifier:

```
mkvdk -collection <collection_name> -optimize spanword
-locale <locale_name>
```

See the *Verity Collection Reference* for more information.

LRC can be invoked for topic creation from Topic Menu and Taxonomy Menu. When LRC is invoked, a progress window will pop out to indicate the current status and progress. You can stop LRC Topic creation by clicking the **stop** button in the window.



If LRC algorithm finds that there are no positive documents during the topic creation, a dialog window will display “No relevant documents for the selected category/topic”.

Invoking LRC From Topic Menu

There are two options for invoking LRC under the Topic pull-down menu:

- **Regenerate Topic Using Document Relevance**
- **Update Topic Using Document Relevance**

These menu options are grayed out (inactive) if any of the following conditions is met:

- The current focus is not in the topic pane.
- No topic is selected.
- The selected topic does not have any relevant documents marked.

To create a new topic using LRC, you can create a topic node first.

Regenerate Topic Using Document Relevance

1. Select an existing topic (or create a new topic node)
2. Flag relevant documents to this topic
3. Flag irrelevant documents to this topic (optional)
4. Choose **Topic | Regenerate Topic Using Document Relevance**.

Update Topic Using Document Relevance

1. Select an existing topic
2. Edit the topic nodes as desired
3. Flag relevant documents to this topic
4. Flag irrelevant documents to this topic (optional)
5. Choose **Topic | Update Topic Using Document Relevance**.

LRC uses the evidence terms in the topic and computes new weights from the exemplary documents.

An existing topic might already have been associated with a set of exemplary documents. To retrieve these flagged documents for a topic, select the topic and choose **Documents | Return Flagged Documents**.

To improve the quality of the generated topics, mark as many irrelevant documents for a topic as possible.

Invoking LRC from the Taxonomy Menu

There are three options under the Taxonomy pull-down menu:

- **Automatically Create Topic From Category**
- **Automatically Create Topic From Category Recursively**
- **Update Topic From Category**

The options are also available from the shortcut menu when you right click a category.

Invoking LRC from the taxonomy menu is quite different from invoking LRC from the **Topic** menu. As discussed in *"Invoking LRC From Topic Menu,"* the latter requires user to explicitly flag exemplary documents to a topic. Only the flagged documents will be used in LRC Topic creation. However, when creating a LRC Topic from categories in a taxonomy, LRC implicitly derives positive and negative documents for a given category from documents associated with the category and negative documents from all other categories in the taxonomy.

A category in the taxonomy pane can be associated with the following lists of documents.

- Documents that are automatically assigned to the category by meta, path, or URL
- Documents that are manually ranked to the category
- Documents that are manually excluded from the category
- Documents that are manually flagged as relevant / irrelevant / non-flag to the category

Auto-associated documents in item 1 are auto-flagged as relevant to the category by default, and can be overwritten by manual operations (ranking, excluding, and flagging). The manually ranked documents are also flagged as relevant documents to the category and are mutually exclusive from the manually excluded documents. Manual flagging (marking) has the highest priority. It overwrites manual ranking and excluding.

Manually ranked documents and excluded documents can be viewed from category property page (Right click on the category and choose **Properties | Manual Ranking**).

When LRC is creating a topic for a category, the union (with overwriting) of the auto-flagged and manually-flagged relevant documents are used as positive documents.

And the negative documents come from the union of

- Documents manually flagged as irrelevant to the category
- All the positive documents (defined above) in all other categories in the taxonomy excluding these documents that appear as positive documents for this category. It is often the case where some documents are assigned to multiple categories in the

taxonomy. The exclusion guarantees that no document can be both positive and negative to a category in the LRC training.

To reduce memory usage, LRC performs a sub-sampling of training documents when the total number of positive and negative documents exceeds 20,000. When this happens, the positive documents of a category are sub-sampled first to obtain up to 10,000 documents (LRC uses every positive document if there are less than or equal to 10,000 positive documents). The negative documents are then sub-sampled and added to the training set to obtain a total of 20,000 training documents.

Taxonomy | Automatically Create | Topic From Category

This option is grayed out if there are no positive documents associated with a category. This option creates an LRC topic and automatically associated this topic in the category. You can view the association in the rule box on the category property page.

Taxonomy | Automatically Create | Topic From Category Recursively

Similar to the previous option, this option creates an LRC Topic for the selected category and every category in the sub-tree rooted at the selected category. LRC Topics are automatically associated with their corresponding categories. The following types of categories are skipped.

- Categories with no positive documents
- Referencing categories

Taxonomy | Update Topic From Category

This option is similar to **Topic | Update Topic From Document Relevance**. One difference is that instead of selecting a topic for updating, you need to select a category. LRC will update the topic associated with the category. This option is grayed out if there is no topic rule associated with the category. Another difference is that the LRC training documents will be derived from categories in the taxonomy.

Preparing Data for LRC

"Creating and Updating Topics Using LRC" discussed where LRC obtains positive and negative documents for each option. This section addresses how to prepare training documents for LRC.

Index Documents with Category Information

Creating LRC Topics from the **Topic** menu requires you to manually flag relevant and irrelevant documents. If you need to create a large number of topics, use the LRC options from the **Taxonomy** menu, because Intelligent Classifier provides a number of methods for automatically associating documents with categories when creating categories from metadata, URL, and file path (see "Creating Categories Automatically"). Intelligent Classifier also provides a method for importing manually ranked documents from a BIF (see Bulk Insert File in the *Verity Command-Line Indexing Reference*) file.

1. If your training documents are in a file system, organize them into a directory structure that reflects the category hierarchy. Once these documents are indexed, the category structure can be easily recovered in Intelligent Classifier using **Taxonomy | Automatically Create | Categories From Path** and choosing **path** (default) in the dialog window. Documents are automatically associated with their corresponding categories.
2. If your training documents are on a web server and category information is already embedded in their URLs, index them using Verity Vspider or K2 Spider (see the *Verity Command-Line Indexing Reference*). You can choose **Taxonomy | Automatically Create | Categories From Path** and choose URL in the dialog window. In both cases (Path and URL), you can enter the prefix-to-ignore to reduce the depth of the taxonomy tree.
3. If the category information of documents is embedded in the documents as metadata, you should index the category information into a VDK field by properly configuring VDK indexing style files (see the *Verity Command-Line Indexing Reference*).
4. If the category information of documents resides in some external media such as in a database, you should retrieve it and convert it into a BIF file and then index the documents with properly configured VDK indexing style files (see the *Verity Command-Line Indexing Reference*).
5. If documents are already indexed and you already have taxonomy defined, you can insert the document-category associations into a BIF file with the same root name as your taxonomy file (.tax), and place this file in the same directory as the taxonomy file. Import this taxonomy (**Taxonomy | Import | Taxonomy**) and establish these

document-category associations. If the BIF is not in the same directory as the taxonomy file or they have different root names, then you can use **Taxonomy | Import | Document Assignments** to establish the associations. For detailed information, see "BIF Files."

How Many Training Documents Needed per Category

The answer to this question is data dependent, that is, how well documents from different categories are distinct from each other. In general, the more documents in the training set, the better quality the resulting LRC Topic will have. You should prepare at least 50 documents per category in order to create a decent topic. Because LRC does subsampling when a category has more than 10,000 positive documents and the total exceeds 20,000 documents, preparing more documents beyond this limit will not help the quality.

It is recommended that you check the evidence terms when a topic is built. If the positive evidence contains too many terms that are irrelevant to the topic based on your knowledge about the subject, that is an indication that you need more training documents for this category and other categories from which the negative documents are derived.

Configuring Stopwords for LRC Topic Creation

To ensure that particular terms do not appear in evidence nodes of LRC topics, you can add these terms to `vdk30.stp` under the corresponding locale directory. You might need to restart Intelligent Classifier after changing `vdk30.stp` in order to make the changes effective.

WARNING! The change in `vdk30.stp` might affect VDK indexing and thematic mapping.

Using LRC From the Command Line

From the command line, you can use mklrc to create a topic from a set of positive and negative exemplary documents or create topics from categories of a taxonomy in a parametric index. The `mklrc` command-line tool is built using the Organization Developer's Kit (ODK).

Note Negative exemplary documents are required for topic creation using `mklrc`, unlike Intelligent Classifier.

You can use `mklrc` to create topics from:

- Lists of documents specified in a BIF file
- Categories in a populated taxonomy in a parametric index
- Auto-extracted categories from a collection using metadata, file path, or URL

mklrc Syntax

The command-line syntax for the `mklrc` tool is shown as follows:

```
mklrc [<option>...]
```

Syntax Descriptions

Table 7-2 describes the `mklrc` syntax elements.

Table 7-2 mklrc Syntax Elements

Element	Description
`-autoextract` *`auto_extract_source`*	The `-autoextract` argument specifies the source from which to automatically extract categories. Must be either `meta`, `path`, or `URL`.
`-autofield` *`vdk_field`*	The `-autofield` argument specifies the VDK field from which to extract categories. You must use with `-autoextract meta`
`-autoflddelim` *`field_delimiter`*	The `-autoflddelim` argument specifies the delimiter to separate categories in the VDK field.
`-autohierdelim` *`hierarchy_delimiter`*	The `-autohierdelim` argument specifies the delimiter to separate categories in the hierarchy.

Table 7-2 mklrc Syntax Elements (continued)

Element	Description
`-bif` *`bif_file`*	The `-bif` argument specifies the BIF file containing relevant and irrelevant documents for the topic.
`-category` *`categories`*	The `-category` argument specifies the categories from which to create topics. Use a space to separate multiple categories.
`-catthresh` *`cat_threshold`*	The `-catthresh` argument specifies the category score threshold. Value can be between 0 and 100. Default value is 50. This value divided by 100 will be used as the threshold that will be associated with categories in the taxonomy.
`-charmap` *`charmap`*	The `-charmap` argument specifies the character map that will be used. The default is the default charmap of the specified locale.
`-collection` *`collection`*	The `-collection` argument specifies the path to the collection that contains the training documents
`-fields` *`field_list`*	The `-fields` argument specifies the VDK collection fields from which `mklrc` will derive evidences in addition to the body text of the documents. Use a space to separate multiple fields.
`-fielddel` *`field_delimiter`*	The `-fielddel` argument specifies the delimiter for parsing the field data. Default value is " " (space). Must have a one-to-one match between delimiters and fields.
`-locale` *`locale`*	The `-locale` argument specifies the locale that will be used.
`-lowerfreq` *`lower_freq`*	The `-lowerfreq` argument specifies the lower threshold of document frequency. Value can be between 0 and 1000. Default value is 30. This value divided by 1000 will be used in LRC evidence selection. A term will not be considered if it appears in less than the specified percentage of the total number of documents in the collections. It is used to filter out terms that are too infrequent to be considered as important. A high value of this parameter means that fewer terms are considered as candidate evidence terms for LRC Topics.
`-lrcinfo`	The `-lrcinfo` argument instructs `mklrc` to return information.
`-lrcpath` *`path`*	The `-lrcpath` argument specifies the LRC working directory where LRC related files will be created and updated.
`-maxevid` *`max_topic_evidences`*	The `-maxevid` argument specifies the maximum number of topic evidences. Value can be between 10 and 1024. Default value is 1024.
`-maxiter` *`max_iterations`*	The `-maxiter` argument specifies the maximum number of iterations in LRC algorithm. Value can be between 20 and 100. Default value is 50.

Table 7-2 mklrc Syntax Elements (continued)

Element	Description
`-notuseexcache`	The `-notuseexcache` argument instructs `mklrc` to not use the existing cache files when processing. The default is to use existing cache files under the LRC path.
`-notuseparent`	The `-notuseparent` argument instructs `mklrc` to not use documents associated with parents as irrelevant documents. Default value is to use documents associated with parents as irrelevant documents.
`-penalty` *`penalty`*	The `-penalty` argument specifies the penalty on misclassification in the training process. Value can be between 1 and 100. Default value is 10. The default value of 10. This value divided by 10 will be used as the penalty value.
`-pipath` *`pi_directory`*	The `-pipath` argument specifies the path to the parametric index directory.
`-prefix` *`prefix_to_ignore`*	The `-prefix` argument specifies the prefix to ignore when using path or URL as the `-autoextract` source.
`-recursive`	The `-recursive` argument instructs `mklrc` to create topics recursively. Applicable only when `-category`, `-tax`, and `-pipath` or `-autoextract` is provided.
`-status`	The `-status` argument instructs `mklrc` to show the current status.
`-tax` *`taxonomy_name`*	The `-tax` argument specifies the taxonomy name from which `mklrc` will implicitly derive positive and negative documents for a given category from documents associated with all the categories in the taxonomy.
`-topic` *`topic_name`*	The `-topic` argument specifies the topic name.
`-topicset` *`topicset_path`*	The `-topicset` argument specifies the path to the topic set.
`-update`	The `-update` argument instructs `mklrc` to update the current topic. The default is to create a new topic.
`-upperfreq` *`upper_freq`*	The `-upperfreq` argument specifies the upper threshold of of document frequency. Value can be between 500 and 1000. Default value is 800. This value divided by 1000 will be used in LRC evidence selection. A term will not be considered if it appears in more than the specified percentage of the total number of documents in the collections. It is used to filter out common terms which cannot discriminate documents from different categories.
`-verbose` *`level`*	The `-verbose` argument instructs `mklrc` to run in verbose mode.

Note For LRC topic creation from categories in a taxonomy, `-category`, `-tax`, and `-pipath` must be provided. In this case the category name with an illegal topic name characters are replaced by and underscore.

LRC Examples

The following command creates two topics from two categories, reuters and earn in taxonomy *taxi* in the parametric index `d:\lrc_data\lrc`.

```
mklrc -category reuters earn
  -collection d:\lrc-data\reut_tr
  -pipath d:\lrc-data\pi\lrcpi1
  -tax tax1
  -topicset d:\lrc-data\tpset
  -lrcpath d:\lrc-data\lrc
```

The following command creates topics recursively from auto-extracted categories from a metafield *TOPICS* in a collection.

```
mklrc -collection d:\colls\reut_test
  -topicset d:\lrc-data\tpset
  -autoextract meta
  -autofield TOPICS
  -autoflddelim " "
  -category Root
  -recursive
```

The following command creates a topic from relevant and irrelevant documents for the topic contained in the specified BIF file.

```
mklrc -locale english
  -fields title
  -collection d:\colls\reut_test
  -topicset d:\lrc-data\tpset
  -lrcpath d:\lrc-data\lrc
  -topic earn
  -bif d:\lrc-data\reuters50.bif
```

The following command shows a section of the BIF file:

```
VdkVgwKey: file1.html
AssignCategory: earn acq
<<EOD>>
VdkVgwKey: file2.html
AssignCategory: crop
<<EOD>>
```

The following command updates a topic from relevant and irrelevant documents located in the specified BIF file.

```
mklrc -locale english
  -fields title
  -collection d:\colls\reut_test
  -topicset d:\tmp\tpset
  -lrcpath d:\tmp\lrc
  -topic earn
  -bif d:\tmp\reuters50.bif
  -update
```

8

Creating Topics Using Hybrid Approaches

This chapter contains the following sections:

Overview of Hybrid Approaches

The following three methods for creating Verity topics with Intelligent Classifier have been discussed in previous chapters:

- Automatic Topic Creation through the LRC (Chapter 7)—automatically derive/enhance topics from positive and negative sample documents
- Business Rules through the use of Verity Topics (Part II)—manually edit existing topics or create new topics based on expert-constructed business rules
- Thematic Mapping (Chapter 5)—automatically generate topics for the key concepts identified in the document set

Each of the three methods provides the user the means to create Verity topics in different circumstances and with different criteria. Automatic topic creation allows automatic generation of highly accurate topics when a sufficient number of positive and negative sample documents is available. Business rule editing allows easy construction and management of critical topics using human intelligence. Thematic mapping allows automatic generation of topics of the key concepts in a document set with no additional requirements.

In addition to using these topic creation methods on their own, you can use them in combination. Combining two or more methods can provide improvement in the quality of the resulting topics or reduce the amount of effort required.

The following diagram shows the data flow between the three methods of topic creation. All three methods produce topics as output, and both Automatic Topic Learning and Business Rule Editing take topics as input.

You can start with any of the three processes and continue with another following any of the possible data flow paths. For example, start with C, and then use (some of) the topics generated by C as input to A, and then use the topics generated by A as input to B. The output of B (the edited topics) can also be used as input to A again, and so on. Possible combinations of the three methods include:

```
C->A->B->A->B ...
C->B->A->B->A ...
A->B->A->B ...
B->A->B->A ...
```

Which Combination Should I Use?

Which combination of the three processes to use depends on a number of factors:

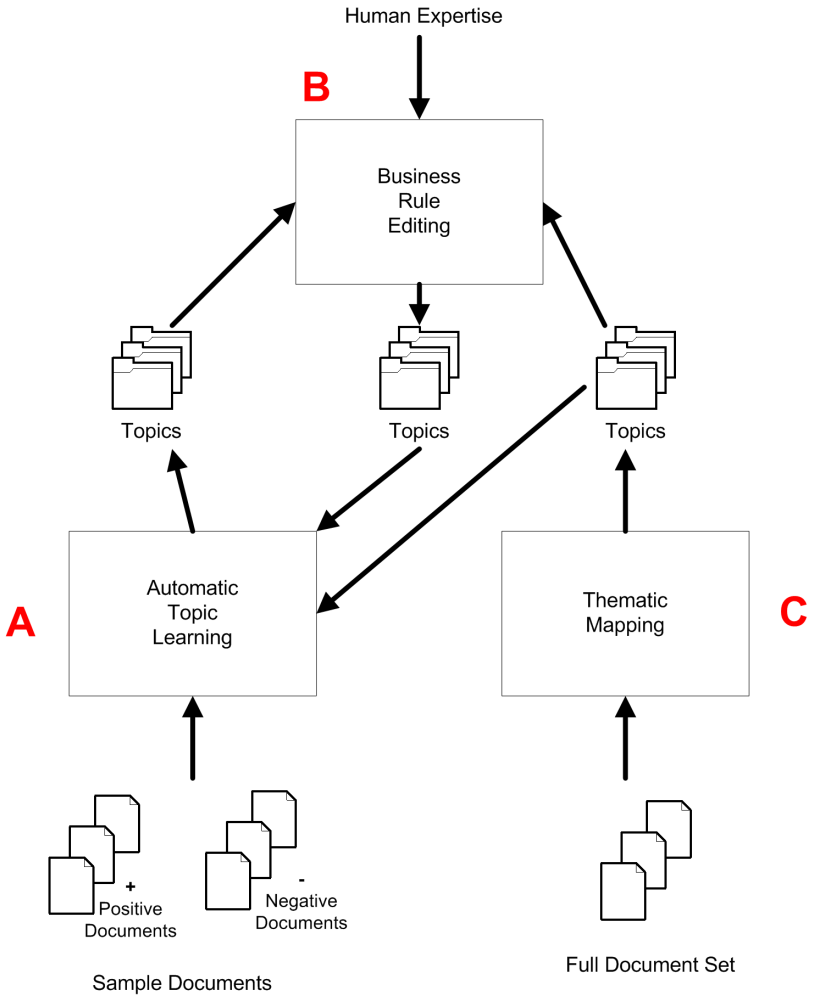
- The accuracy of the topic you require
- The amount of time and effort you want to spend
- The amount of human expertise you have available
- The amount of positive and negative training data (sample documents) you have available

For example, if sufficient human expertise and time are available, you might want to manually construct the initial set of topics based on Business Rules (B), and then enhance these topics with Automatic Topic Learning using some positive and negative sample documents (A).

If, instead, a large amount of positive and negative training data is available, the user might want to start with Automatic Topic Learning (A), and then manually review and edit the topics generated (B).

If little human expertise, time, and positive and negative training data are available, you might want to start with Thematic Mapping (C), review the generated topics and select the ones that are relevant, and then enhance these selected topics using Automatic Topic Learning (A) or Manual Editing based on Business Rules (B), or both.

Figure 8-1 Topic Creation Using Hybrid Processes



Other Hybrid Approaches

In addition to the work paths suggested above, Intelligent Classifier allows many more methods of interaction between the three topic creation processes. For example, a user can first use Thematic Mapping (C) to automatically create some topics, conduct searches on these topics to locate a set of candidate training documents for each of the topics, and manually tag a portion of these candidate training documents sets as positive and negative sample documents, and finally use these tagged documents as input training data for Automatic Topic Learning (A).

In another possible work path, a user can manually construct some topics based on Business Rules (B), conduct searches on these topics to select a portion of the document set, and then use Thematic Mapping (C) to generate new topics from the key concepts found in the subset of documents.

Using VQL Operators with Hybrids

You can use different sets of VQL operators and modifiers in the topics created through the three different methods. LRC uses the <LOGSUM> operator, and topics created from Concept Mapping use the <ACCRUE> operator, while manually created topics can use all the operators supported in VQL. When a topic created from a concept is refined by LRC using a set of positive and negative documents, the new topic will be in the LRC topic format.

Similarly, refining a manually created topic using LRC will yield a new topic in the LRC topic format, no matter what operators and structure are in the original topic.

A refined topic might not be necessarily better than the original topic. This largely depends on the quality and amount of information available in the refining process. Proper preparation of data and testing after a refinement is strongly recommended.

PART III

Populating Taxonomies

Part III discusses how to create and update parametric Indexes from taxonomies in Intelligent Classifier, and how to build and maintain knowledge trees from the command line for use in a K2 Search System.

The following chapters are included:

- Parametric Index Creation
- Building Knowledge Trees From the Command Line

9

Parametric Index Creation

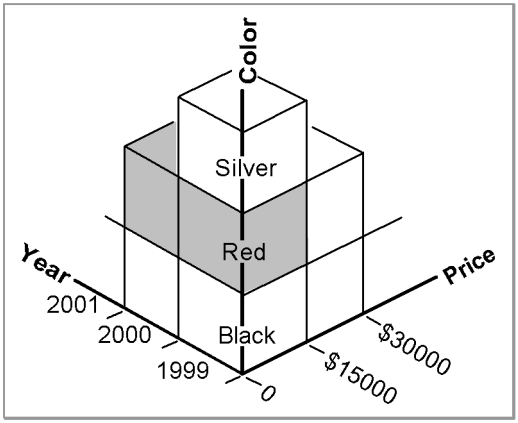
This chapter contains the following sections:

Introduction to Parametric Indexes

A *parametric index* is an index that allows retrieval of documents based on the values of parameters. Parametric indexes are the structures that underlie parametric search. Think of a parametric index as an extension to a collection's word index; the Verity engine uses it to identify documents matching the requested parameters.

You can view a parametric index as an n-dimensional "parametric cube" in which each dimension represents a parameter. Parameters are values that can be categorized into useful groupings or sets. For example, Figure 9-1 shows a portion of a three-dimensional version in which the parameters are car color, model year, and price.

Figure 9-1 Parametric Cube



Each individual value or range of values for a parameter is called a *bucket*, and each parameter as a whole (each dimension of the cube) makes up a *bucket set*. Each bucket holds references to the documents that have that value for that parameter. For example, in the portion of the cube shown in Figure 9-1, the **Color** bucket set contains three buckets: **Black**, **Red** and **Silver**. The **Red** bucket identifies all documents in the collection that relate to red cars.

In this example, a parametric selection for all red cars costing less than $15,000 made in either 1999 or 2000 (the shaded area in Figure 9-1) returns only those documents with field values that satisfy all three criteria.

Bucket parameter values are represented as:

- Enumeration—Each unique value is associated with its own bucket in a bucket set. For example, keys for each unique color, such as red, blue, and white, are stored in separate buckets.

- Trees—Values are associated with categories in a taxonomy. The taxonomy specification might be pre-existing, or it can be created dynamically based on fields in the collection or XML data.

  In addition, bucket sets can be defined based on an XML format taxonomy file. See "Importing and Exporting Taxonomy Files" for information on how to create these.

- Attributes—A dynamically-created bucket, consisting of a name-value pair. The name is the dynamically-created bucket. The value is the value associated with that bucket. A bucket set is the set of all attribute buckets for a specified collection or XML field.

For additional information about parametric indexes, see the *Verity K2 Parametric Developer Guide*.

Using Intelligent Classifier to Create a Parametric Index

With Intelligent Classifier, you can create a parametric index based on fields in a collection or categories from multiple taxonomies. XML elements provide extra fields for documents in a collection.

Because a parametric index directly supports an Intelligent Classifier taxonomy (functionally equivalent to the `.tax` representation of a taxonomy), you can create parametric indexes that contain tree type bucket sets that correspond to specific taxonomy categories. The default names of the bucket sets are the starting category name of those categories. Each bucket set's enumeration values correspond to all category nodes under the set, including its own.

Intelligent Classifier loads only one taxonomy at a time, so you cannot directly build relational taxonomies based on multiple taxonomies. To do this, you must create an empty taxonomy in Intelligent Classifier and import the multiple taxonomies as subtrees under the root node of the empty taxonomy. Then you can select the root nodes of those subtrees and create the parametric index.

Creating a Parametric Index

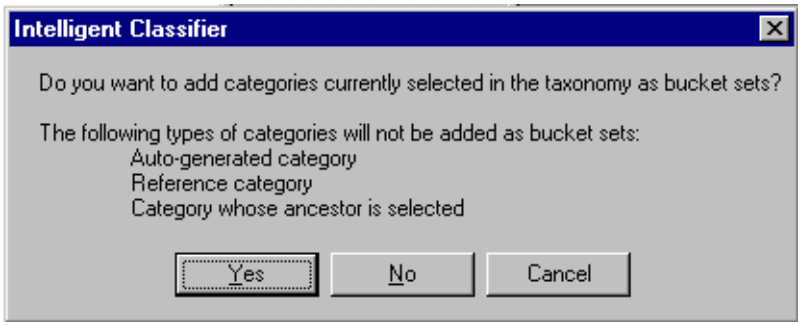
This section describes how to create parametric indexes from Intelligent Classifier. Parametric Indexes can also be created from the command line with `mkpi`. For information on how to do this, see the *Verity K2 Parametric Developer Guide*.

This section shows you how to create a parametric index from a taxonomy. You must have a collection loaded to do this.

1. Choose **File | Create Parametric Index**.

   If you have selected categories in the current taxonomy, the following dialog appears:

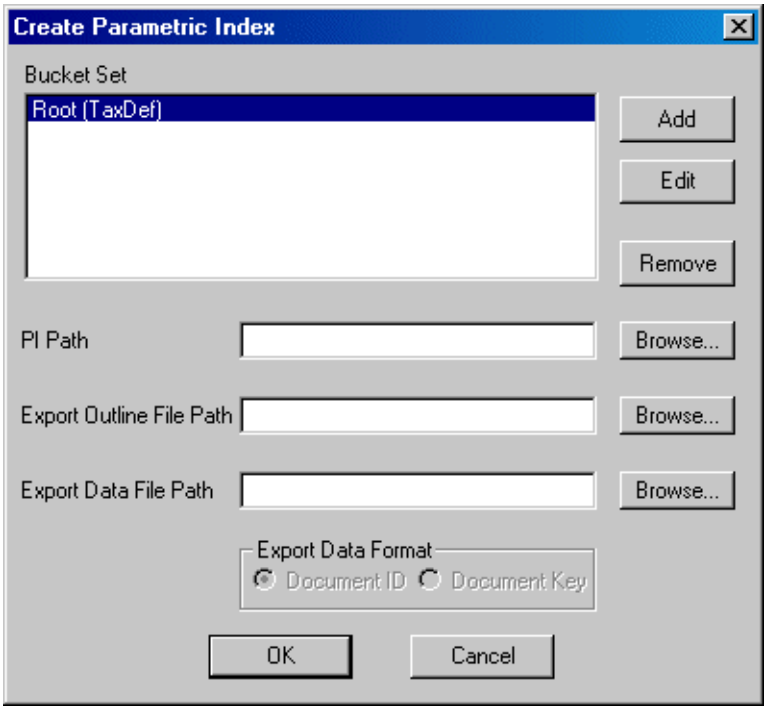
Figure 9-2 Add Categories dialog



2. Click **Yes** to add the selected categories as buckets sets in the parametric index.

   The **Create Parametric Index** dialog appears.

Figure 9-3 Create Parametric Index dialog



Items that appear in the bucket set list have already been defined. The text in the parenthesis following each bucket set name indicates its type.

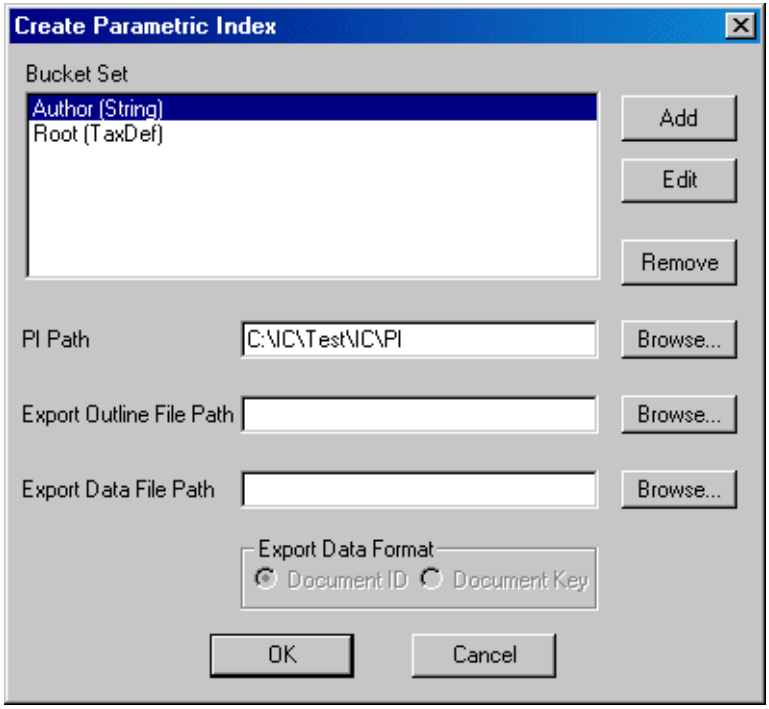
TaxDef indicates that the bucket set is based on a category in the current taxonomy.

If you had categories selected in the taxonomy, and chose **Yes** for the dialog in Figure 9-2, the selected categories appear in this list.

Populating the Parametric Index

1. In the **PI Path** field, specify a location for the parametric index by typing in the path or using the **Browse** button to navigate to the location.

Figure 9-4 Bucket Set List dialog



2. Specify a location for the export outline and data files, if required.
   - Specify **Export Data File Path** to create the specified files during Parametric Index generation in a format that can be used by `mkpi` to create a Parametric Index from this exported XML-Data file.
   - If **Export Outline File Path** is specified but **Export Data File Path** is left blank, then the specified outline file is created in the original format to create the collection-based Parametric Index from Intelligent Classifier.
   - Specifying paths to both files creates an XML Data outline file.

3. Choose the export data format:
   - **Document ID**—specifies that the exported data file identifies documents with a numeric ID.
   - **Document Key**—specifies that the exported data file identifies documents with the document VdkVgwKey.
4. Click **OK**.

   The cursor changes to the hour glass during the populating process. When the parametric index creation has completed, a dialog appears stating you have successfully created a Parametric Index.

Note When you create a parametric index, Intelligent Classifier saves temporary parametric index outline (*pi_name*`.tmp.xml`) and taxonomy (*taxonomy_name*`.tmp.xml`) files in the IC directory.

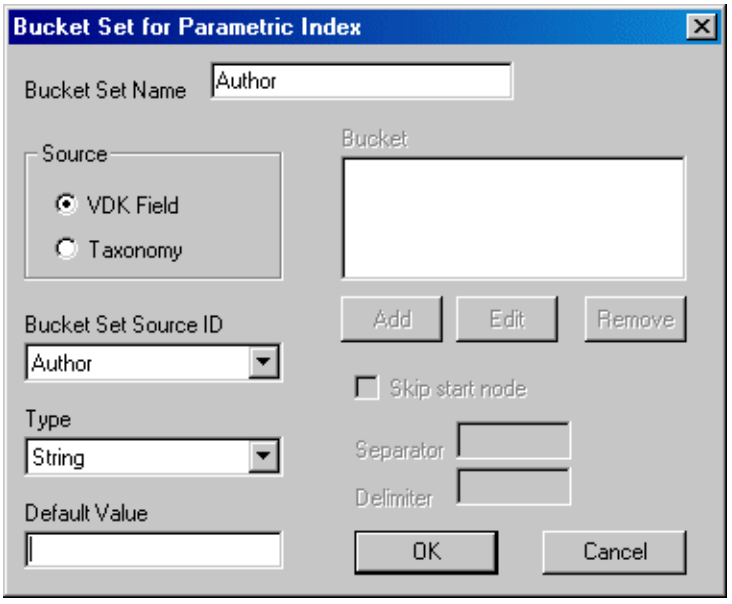
Adding a Bucket Set from a Collection Field

In the following steps, you add a new bucket set to your parametric index based on a field in your collection.

1. In the **Create Parametric Index** dialog, click **Add.**

   The **Bucket Set for Parametric Index** dialog box is displayed for you to define a new bucket set

Figure 9-5 Bucket Set for Parametric Index dialog



2. In the **Bucket Set Name** field, type *Author* as the name for the bucket set.
3. In the **Source** group of options, select **VDK Field**.

   The following options indicate where the bucket set information will come from.

   - **VDK Field** indicates the bucket set is based on a collection (VDK) field.
   - **Taxonomy** specifies that the bucket set is based on a category from a taxonomy. If you select **Taxonomy**, the **Skip start node** check box is selected by default. Selecting the **Skip start node** check box removes the bucket set root node and its corresponding path component in the category enumeration.
4. Select *Author* from the **Bucket Set Source ID** drop-down list box.

   Author is the name of the field in the collection on which to base the bucket set. The list box shows all available fields in the collection.
5. Select **String** from the **Type** drop-down list to specify the data type of the bucket set.

   The **Type** drop-down list box only applies to **VDK Field** type bucket sets.

   The **Default Value** field also applies to **VDK Field** bucket sets. If you leave the field empty, Intelligent Classifier does not set a default value for a bucket set.
6. Select **OK**. The **Create Parametric Index** dialog now displays the newly created *Author* bucket set in the **Bucket Set List**.

Adding a Bucket Set from a Category

In the following steps you will add a new bucket set to your parametric index that will be based on a category in your taxonomy.

1. In the **Create Parametric Index** dialog, click **Add.**

   This brings up the **Bucket Set for Parametric Index** dialog box where you can define a bucket set.

2. In the **Bucket Set Name** field, type `Author` as the name for the bucket set.

3. In the **Source** group of options, select **Taxonomy**.

4. Select `Author` from the **Bucket Set Source ID** drop-down list box.

   `Author` is the category ID in the taxonomy on which to base the bucket set.

   The list box shows all available fields in the collection.

Types of Bucket Sets

There are other types of bucket sets for VDK fields that you can add to the parametric index. Table 9-1 contains a complete list of types you can select from the **Type** list box in the **Bucket Set for Parametric Index** dialog box.

Table 9-1 Bucket Set Data Types for Parametric Indexes

Data Type	Description
Attribute	A string that specifies a name-value pair.
Date	A string in the date export format. By default, the format is: MM-DD-YYYY HH:MM:SS.
Float	A real number in IEEE floating-point representation.
Int	A 4-byte signed number in the range of -2,147,483,648 to +2,147,483,647, inclusive.
String	A null-terminated character string.
Tree	A string that specifies a category in a hierarchal organization of categories.
Unsigned	A 4-byte unsigned number in the range of 0 to 2^{32}-1.

Buckets

Numeric bucket set types allow you to define one or more buckets. These types are:

- Int
- Date
- Unsigned
- Float

A bucket is a value or range of values within a bucket set. For instance, the bucket set *Age*, could have a bucket *Minor* that has a range <19, and another bucket *Adult* with a range of 19-55, and a final bucket *Senior* with a range >55.

Adding a Date Bucket Set

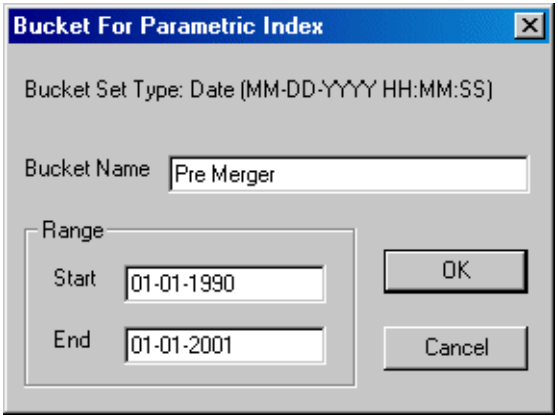
In the following steps you will add a date type bucket set linked to a VDK field to a parametric index. You will also define a bucket with a date range.

1. In the **Create Parametric Index** dialog, click **Add.**

   This brings up the **Bucket Set for Parametric Index** dialog box where you can define a bucket set.

2. In the **Bucket Set Name** field, enter `Version` as the name for the bucket set.
3. In the **Source** group of options, select **VDK Field**.
4. Select `Version-Date` from the **Bucket Set Source ID** drop-down list box.
5. Select `Date` from the **Type** drop-down list to specify the data type of the bucket set.
6. Click **Add**.

   This brings up the **Bucket for Parametric Index** dialog:



7. In the **Bucket Name** Field, enter `Pre Merger`.
8. Under **Range**, enter `01-01-1990` in the **Start** field.

   The **Start** value is inclusive, meaning from, and including the start date.
9. Enter 01-01-2001 in the **End** field.

   The **End** value is exclusive, meaning up to, but not including the end date.
10. Click **OK**.

    The new bucket will now appear in the **Bucket** list box of the **Create Parametric Index** dialog. You can create more buckets for the bucket set as needed.

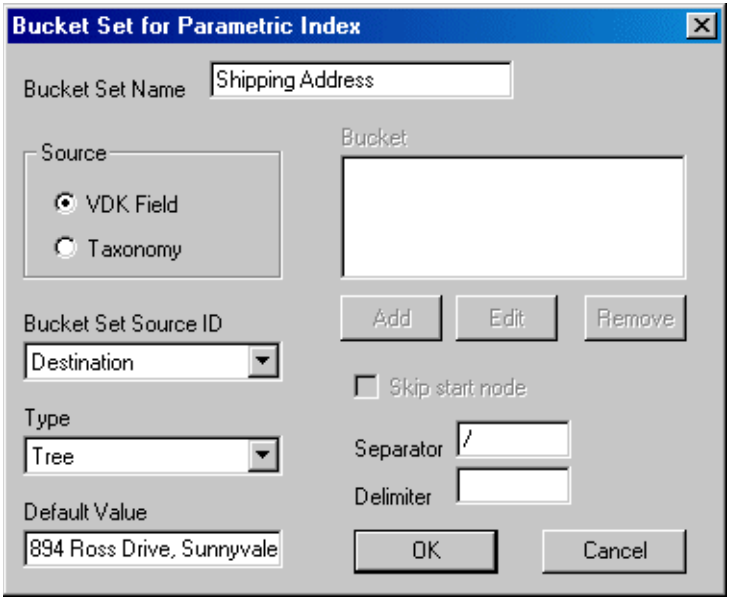
Adding a Tree Bucket Set From a VDK Field

In the following steps you will add a taxonomy type bucket set linked to a VDK field.

1. In the **Create Parametric Index** dialog, click **Add.**

   This brings up the **Bucket Set for Parametric Index** dialog box where you can define a bucket set.

Figure 9-6 Adding a Tree Bucket Set from a VDK field



2. In the **Bucket Set Name** field, enter `Shipping Address` as the name for the bucket set.
3. In the **Source** group of options, select **VDK Field**.
4. Select `Destination` from the **Bucket Set Source ID** drop-down list box.
5. Select `Tree` from the **Type** drop-down list to specify the data type of the bucket set.
6. Enter / in the **Separator** edit box.

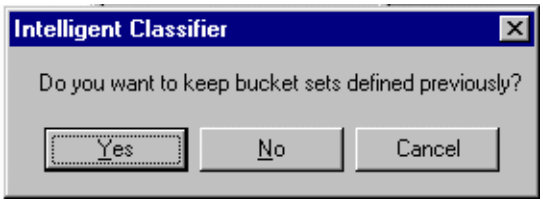
   The separator string separates the hierarchy levels of the tree; in this example, it is the forward slash (/) character. You can also use them with `Attribute` type bucket sets.

   You can also enter a character or string into the **Delimiter** field to distinguish multiple categories. This string is different from the separator string.
7. In the **Default Value** field, enter: `894 Ross Drive, Sunnyvale, CA 94089`.
8. Click **OK**

   The new bucket appears in the **Bucket Set** list box of the **Create Parametric Index** dialog. You can create more bucket sets as needed.

Persistence of Bucket Set Definitions

When you create a parametric index, the bucket sets defined for it persist in the Intelligent Classifier session. When you start to create a new parametric index in the same Intelligent Classifier session, the following dialog box appears:



- Selecting **Yes** adds previous bucket set definitions to the new parametric index.
- Selecting **No** clears previous bucket set definitions and does not add them to the new parametric index.

If a workspace or knowledge tree has defined bucket sets from the creation of a parametric index (**File|Create Parametric Index**), the current bucket set definitions are saved in the workspace file when the workspace or knowledge tree is closed.

When you open a workspace or knowledge tree containing bucket set definitions, the bucket set definitions in the session are replaced with the bucket set definitions from that workspace.

When you open a workspace or knowledge tree without bucket set definitions, the bucket set definitions from the session are retained.

10

Building Knowledge Trees From the Command Line

This chapter describes how to build and maintain Knowledge Trees for use in the K2 Search System using command–line tools. Knowledge Trees can also be created from Intelligent Classifier. For information on how to do this, see "Intelligent Classifier Concepts" on page 27.

The following topics are covered in this chapter:

Understanding the Command-line Tools

You can build a hierarchical set of categories called a Knowledge Tree that organizes the documents indexed into a collection. If the collection is considered an index of words in the documents, then the Knowledge Tree can be considered the table of contents that organizes the documents into subjects of interest.

You can build categories automatically from the data stored in the collection. Administrators can base their categories on a file system directory structure, Web site URL organization, the values in document fields (for example, subject field in WYSIWYG documents or META tags in HTML documents), Verity topic sets, and other queries. The Knowledge Tree structure is stored in the following databases:

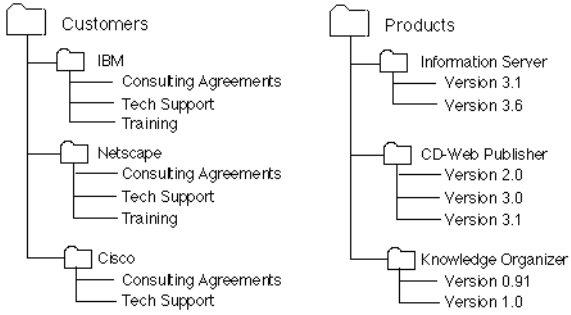
- The Taxonomy database (.TAX file) stores the hierarchical structure of the categories.
- The Category Index database stores the relationship between categories and the documents assigned to them.

The category hierarchy helps users both pinpoint relevant information in familiar categories and discover new areas of interest in related categories. Users do not need to know all of the relevant words in the collection to locate information; categories guide users to other relevant document groupings.

Using the Knowledge Tree to Organize Documents

A Knowledge Tree applies a *category capture* method to documents indexed into a Verity collection to extract *categories* that organize the documents into subjects of interest. A Knowledge Tree consists of a *taxonomy* of browsable categories and databases that stores the relationship of documents and categories.

More than one Knowledge Tree can be associated with a collection to provide different views of the same document set. For example, if the collection contains Verity's correspondence, one Knowledge Tree organizes the documents by customer while the other organizes them by product.



Capturing Categories for Use in Knowledge Trees

You can use several category capture methods to categorize documents into a Knowledge Tree:

- Extract categories from URL paths
- Extract categories from file paths
- Fetch categories from a collection field
- Compute categories by applying classification rules (queries) to the document
- Look up manual category assignments made by the administrator

You can use the collection field method when the categories are explicitly listed in a field in the collection. This is the case, for example, when categories are specified in a META tag inside HTML documents that gets extracted into a collection field during indexing. The category IDs are simply parsed out of the field.

You can use the URL and file path methods to extract a document's category from its URL or file path listed in a collection field. New categories are automatically added to the taxonomy as they are extracted. This method enables administrators to mirror the web site or file system structure in the Knowledge Tree taxonomy.

You can use the classification rule method to assign categories to documents based on classification queries defined for the categories. All categories whose associated query matches a document are assigned to the document.

You can use the manual assignment method to augment one of the other capture methods, applying category assignments made by an administrator to override field-based or rule-based assignments.

Using Taxonomies to Organize Categories

A taxonomy organizes a set of categories, and the structure and definition of the hierarchy is stored in a taxonomy database. A taxonomy database can be shared by one or more Knowledge Trees, or it can be created specifically for a single Knowledge Tree. When a new Knowledge Tree is created, the default is to construct a new taxonomy database specifically for that Knowledge Tree.

Category Properties

Categories organize documents into subjects of interest and the category-to-document relationships are stored in a category index database.

A category must have a unique ID to identify it within the taxonomy. You use category IDs to specify categories for browsing and scoping in the browse server. You also use IDs to specify categories to assign to documents when building the Knowledge Tree, such as in document META tags, collection fields, or manual submissions to the Knowledge Tree.

In addition to an ID, a category can have user-defined properties. Two of the important properties are:

- Name—The label presented for the category in the navigation interface. While the category ID can be terse or cryptic, the category name should be a short label that is meaningful to the end user.
- Description—The optional long description of the category that can be shown in the user interface.

Knowledge Tree Management Tool (ktmgr)

Use the Knowledge Tree management tool, `ktmgr`, to create and maintain Knowledge Trees from scripts or from the command line.

The basic syntax is:

```
ktmgr -kt path [option...]
```

Where *`path`* is the path to the Knowledge Tree and `[option...]` can be any of the options listed below. The `-k2` option must be used to create or update a Knowledge Tree for use in a Verity application. The `-k2` option builds an optimized Knowledge Tree—a Knowledge Tree that can be searched by K2 Servers.

An optimized Knowledge Tree must be updated using `ktmgr` with the `-k2` option when a Knowledge Tree itself has been updated. If the taxonomy or associated document collection have been updated, `ktmgr` must be run with the `-k2` option to support Knowledge Tree searching.

ktmgr Syntax Reference

The following sections list the ktmgr options grouped by the activity where they are most frequently used, and then alphabetically. Descriptions of each options are in Table 10-5.

ktmgr Options Grouped by Activity

Table 10-1 through Table 10-4 list the `ktmgr` options by the activity.

Table 10-1 Create Knowledge Tree Activities

Create Knowledge Tree	Import a Taxonomy File	Populate Knowledge Tree
`-kt` *path*	`-kt` *path*	`-kt path`
`-collection` *string*	`-importtax` *file*	`-filter` *query*
`-create`	`-mergetax`	`-persist` *polling-interval-secs*
`-kb` *path*		`-populate` *mode*
`-style` *path*		`-credentials` *string*
`-taxonomy` *path*		`-topicset` *path*

Table 10-2 Recategorize Knowledge Tree Activities

Recategorize Knowledge Tree	Reindex Knowledge Tree	Index Knowledge Tree Taxonomy
`-kt` *path*	`-kt` *path*	`-kt` *path*
`-filter` *query*	`-reindex`	`-indextax`
`-persist` *polling-interval-secs*	`-indextax`	
`-recategorize` *mode*		

Table 10-3 Export Taxonomy File Activities

Export a Taxonomy File	Remap Knowledge Tree	Common Options
`-kt path`	`-kt path`	`-vdkhome path`
`-exporttax file`	`-remap`	`-charmap file`

Table 10-4 Bulk Submit Activities

Bulk Submit	Request Debugging Output	Knowledge Tree Maintenance
`-kt path`	`-kt path`	`-kt path`
`-autodel`	`-debug options`	`-backup path`
`-bulk`	`-verbose options`	`-optimize type`
`-delete keys`		`-service`
`-edit keys`		`-purge`
`-insert keys`		
`-numdocs spec`		
`-offset num`		
`-update keys`		

ktmgr Options Reference Table

Table 10-5 provides an alphabetical list of the `ktmgr` options.

Table 10-5 ktmgr Options Listed Alphabetically

Option	Description
`-autodel`	Deletes the bulk file(s) after the documents are submitted to the directory.
`-backup path`	Backs up the Knowledge Tree database to the given new path. Safely copies the internal indexes of the database so that if they are copied while an indexing operation is taking place the backup is left in a consistent state. This option is useful for safely copying Knowledge Trees to new locations.
`-bulk`	Specifies that the keys listed for `-insert`, `-update` or `-delete` are bulk submit files containing a list of entries to submit.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
`-charmap` *`path`*	Specifies the character map used for importing and exporting TAX files. If the TAX file is in a character set that is not the default for the locale, you must specify `-charmap`. `-charmap` can also be used to import or export TAX files in a specific character set.
`-collection` *`string`*	Specifies the collection to which the Knowledge Tree is attached. Use this option when creating a new Knowledge Tree.
`-k2`	Creates an optimized knowledge tree that enables searches with Verity K2 Search. This option creates an additional knowledge tree component (data file) that is required to be created and maintained in a K2 Search System. The data file is stored in: `treepath/index/category.vct` where `treepath` represents the path to the knowledge tree. `ktmgr` stores ranking information to the `.vct` file. This means that that ranking can be combined with other fields to sort documents over a single knowledge tree.
`-create`	Creates a new Knowledge Tree at the specified path. Use this option to specify `style` path and `collection` when creating a new Knowledge Tree.
`-debug` *`options`*	Requests debugging output. Specify `vdk`, `tax`, or both to get debug-level output from the Verity search engine or the Knowledge Organizer engine, respectively.
`-delete` *`keys`*	Deletes the specified keys from the Knowledge Tree. The keys are interpreted as bulk submit files if the `-bulk` flag is specified.
`-edit keys`	Edit is similar to `-update`, but is used to modify the manual categorization fields or other annotation fields for documents without reapplying the category capture methods. The `-edit` option works only with bulk submit files. For example, to change the Annotation field for a document, use the `-edit` option with a bulk submit file that contains the document key along with the new Annotation field. The current categorization of the document is preserved. To add or remove a category from a document manually, use a bulk submit file containing the document key along with the updated assignment or unassignment fields (as specified in the `style.mcd` file). The current categorization is modified based on the new manual assignments or unassignments.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
`-exporttax` *file*	Exports the categories in the Knowledge Tree's taxonomy to the specified `TAX` file.
`-filter` *query*	Specifies a categorization filter. This is used in conjunction with `-populate` and `-recategorize` to restrict the set of documents to be categorized. The filter can be any Verity query against the collection. Only the candidate documents matching the filter are submitted to the Knowledge Tree. With a filter you can populate a Knowledge Tree from only a subset of a collection, which is useful, for example, in fine-tuning a set of categorization rules on small document sets before categorizing a large collection.
`-importtax` *file*	Imports new categories into the Knowledge Tree's taxonomy from the specified `TAX` file. The existing categories in the Knowledge Tree's taxonomy are deleted before importing the new ones unless the `-mergetax` flag is specified.
`-indextax`	Builds a full-text index of the names and descriptions of categories in the Knowledge Tree's taxonomy. This index is used by the navigation server to find categories whose names or descriptions match the query. This option updates the taxonomy index. To accomplish the equivalent of clicking the Reindex button, you must also use the `-reindex` option.
`-insert` *keys*	Inserts the specified keys into the Knowledge Tree, capturing and indexing their categories as they are submitted. The keys must be `VdkVgwKeys` that exist in the primary collection unless the `-bulk` flag is given, in which case the keys are interpreted as bulk submit files containing the keys to submit. The bulk submit file should contain `VdkVgwKey` entries that already exist in the collection. In the bulk submit file, if the entry for a document includes a Category field, the categories in that field are used as the categories for the document, overriding the category capture methods for that document. This allows category assignments to be manually forced into the Knowledge Tree entirely through bulk submit files. See the *Verity Collection Reference* for the bulk submit file syntax.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
`-kb` *path*	Specifies the path to the knowledge base (`.kbm` file) defining topics used in the classification rules for the Knowledge Tree. This is optional when creating a new Knowledge Tree. A pointer to the knowledge base is saved in the Knowledge Tree so it can be reopened whenever the Knowledge Tree is opened for update. When this option is used, `ktmgr` does not index the topic set.
`-kt` *path*	Specifies the path to the knowledge tree (required).
`-locale` *string*	Specifies the locale that is used. The locale used must match the locale of the document collection associated with the knowledge tree.
`-mergetax`	Specifies that the taxonomy being imported using the `-importtax` option is to be merged into the existing taxonomy rather than replacing it. Adds new categories to the existing taxonomy. Categories that already exist in the taxonomy are updated with any new definitions in the import file.
`-numdocs` *num*	Specifies the number of documents to submit from the bulk file. Only meaningful if the `-bulk` flag is set and there is only a single bulk file listed for `insert`, `update`, or `delete`.
`-offset` *num*	Starts submitting from the given offset (in bytes) into the bulk submit file. Only meaningful if the `-bulk` flag is set and only a single bulk file is listed for `insert`, `update`, or `delete`.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
-optimize *types*	Optimizes the category index in the Knowledge Tree database. Specify one or more of the following optimization types separated by spaces: ■ `all`—Performs a full optimization, running all of the individual optimization modes. ■ `merge`—Performs maximum merging of partitions in the category index. This mode optimizes search performance. (See the *Verity Collection Reference* for more information on partition merging.) ■ `squeeze`—Squeezes deleted documents from the category index. Optimizes disk usage by reclaiming space taken up by documents that are marked as deleted but not yet removed from the database. (See the *Verity Collection Reference* for more information on squeeze.) ■ `verify`—Verifies that each entry in the Knowledge Tree has a corresponding entry in the primary collection, deleting entries that have no mapping. "Dead" entries in the Knowledge Tree can occur when documents are deleted from the collection after being submitted to a Knowledge Tree. Knowledge Tree entries without a corresponding document in the collection do not affect search behavior—they are ignored—but they should be periodically removed to recover the space and improve search performance. This optimization type deletes all "dead" entries and then squeezes them from the index. ■ `reorder`—Rearranges the records in the Knowledge Tree to optimize the performance of scoped searches. This mode arranges the records in the order that makes the database join required for scoped searches most efficient. Run this mode only after making many manual insertions to the Knowledge Tree, or after incrementally synchronizing the Knowledge Tree to the collection several times when many of the collection's documents have been updated. In environments with frequent updating of the Knowledge Tree, where documents are being frequently updated and deleted from the collection, you should periodically run a full optimization to recover all dead space from the index and to optimally order the records.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
-persist *polling-interval-secs*	Used in conjunction with the `delta` modes of `-populate` and `-recategorize`, this option specifies that the system should populate or recategorize persistently, polling the collection for changes every *polling-interval-secs*. This option automatically updates the Knowledge Tree as the collection changes. The program runs until you stop it.
-populate *mode*	Populates the Knowledge Tree with documents from the attached collection. This is used to categorize all documents in the collection, automatically keeping the Knowledge Tree synchronized with the collection. Specify one of the following modes: ■ `full`—Submits all documents from the collection for categorization in the Knowledge Tree. The usual category capture and indexing process occurs as documents are entered into the Knowledge Tree. Documents from the collection that already have entries in the Knowledge Tree will be recategorized and their entries will be updated. ■ `delta`—Submits only documents from the collection that have been added or updated since the last population. This mode keeps a Knowledge Tree synchronized with a collection as it changes over time.
-purge	Removes all documents from the Knowledge Tree, restoring the Knowledge Tree to the state it was in when it was first created. Only the documents are purged; the taxonomy remains intact.
-recategorize *mode*	Recategorizes documents in the Knowledge Tree. This is used to update the categorizations for documents already in the Knowledge Tree, keeping the Knowledge Tree synchronized as classification rules change or as the documents are updated in the collection. This option enables you to keep a Knowledge Tree that contains only a subset of the documents from the attached collection current without adding any new documents. Specify one of the following modes: ■ `full`—Submits all documents from the Knowledge Tree that exist in the collection for recategorization. The category capture and indexing process occurs as documents are submitted to the Knowledge Tree. ■ `delta`—Submits only documents from the Knowledge Tree that have been updated in the collection. Use this mode to keep a Knowledge Tree current as documents are updated in the collection. No new documents are added.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
-reindex	Reindexes the documents in the Knowledge Tree to reflect changes to the taxonomy structure. The Knowledge Tree keeps an index of both the documents that are strictly within each category and the documents that are recursively within each category (in any subcategory of a category) in order to accelerate scoped search. When the hierarchical category structure changes, the "recursive" index becomes invalid, so the documents have to be reindexed against the new hierarchical structure. Run this option whenever the existing taxonomy structure is modified so that users can obtain accurate scoped search results. This option updates the category index. To accomplish the equivalent of clicking the Reindex button, you must also use the `-indextax` option.
-remap	Remaps the Knowledge Tree to the collection. This option is used to update the Knowledge Tree-to-collection ID mapping for documents in the Knowledge Tree that have been updated in the collection. The mapping must be kept current so that the scoped search and grouping features to work properly. This command enables you to keep a Knowledge Tree synchronized with a collection as documents are updated without recategorizing the updated documents. This is useful, for example, if you have built a Knowledge Tree through manual submission with explicitly specified categories and wish to keep the Knowledge Tree synchronized with the collection without losing the original categorization.
-service	Services the category index in the Knowledge Tree, giving the indexing engine time to perform self-service maintenance operations. This is generally not required.
-style *path*	Specifies the path to the style directory containing the configuration and schema files for the Knowledge Tree. Specify the `-style` option when creating a new Knowledge Tree. The default location of the style directory is `<install_dir>\k2\common\styles\tcstyle`.

Table 10-5 ktmgr Options Listed Alphabetically (continued)

Option	Description
-taxonomy *string*	Specifies the path to the taxonomy database to use in the Knowledge Tree. This is optional when creating a new Knowledge Tree. If no taxonomy is specified, a new, empty taxonomy is created within the Knowledge Tree. This option is typically used to have multiple Knowledge Trees share a common taxonomy database. `-taxonomy` does not take `TAX` files for its argument; it uses a taxonomy database directory that was created using `taxmgr`. `-taxonomy` is typically used for sharing a single taxonomy database among several Knowledge Trees. Ensure you fully understand the `-taxonomy` option before using it. To import categories from an existing `TAX` file, you must use the `-importtax` option.
-topicset *path*	Specifies the path to the topic set you want to use with the collection to which the Knowledge Tree is attached. This option is ordinarily not required when populating a new Knowledge Tree. The `-topicset` option does not index the topic set. This option *is* required when populating a new Knowledge Tree using a taxonomy file that has been created by the `top2tax` command line tool. Does not perform topic indexing in the knowledge tree, making knowledge tree generation with `-topicset` faster than in releases prior to 4.5.
-update *keys*	Updates the specified keys in the Knowledge Tree, capturing and indexing their categories as they are submitted. If the keys already have entries in the Knowledge Tree, they are replaced by the new entries. The keys are interpreted as bulk submit files if the `-bulk` flag is specified.
-vdkhome *path*	Specifies the Verity common directory.
-verbose *options*	Requests verbose output. Specify `vdk`, `tax`, or both to get verbose output from the Verity search engine or the Knowledge Organizer engine, respectively.

ktmgr Examples

This section contains examples of how you use the `ktmgr` command.

Create a new knowledge tree and populate it with documents

1. Create a Knowledge Tree attached to the `testcoll` collection:

   ```
   ktmgr -create -kt d:\kts\mykt -style dirstyle_copy -collection
   d:\colls\1 -importtax d:\taxonomy\newtax.tax -kb d:\kts\
   topicset.kbm
   ktmgr -kt d:\kts\mykt -populate full -topicset d:\kts\
   topicsetdir
   ktmgr -kt d:\kts\mykt -k2
   ```

   Where:

   - *d* is the drive where the knowledge tree will be created.
   - `mykt` is the name of the new knowledge tree.
   - *newtax.tax* is the name of the taxonomy to be imported into the knowledge tree.

2. Import an existing taxonomy that has classification rules and build the index for category names and descriptions:

   ```
   ktmgr -kt testkt -importtax mytax.tax -indextax
   ```

3. Categorize all documents in the collection and generate k2 component:

   ```
   ktmgr -kt testkt -populate full
   ktmgr -kt testkt -k2
   ```

Modify classification rules and recategorize all documents

1. Export the Knowledge Tree's taxonomy to a file:

   ```
   ktmgr -kt testkt -exporttax testkt.tax
   ```

2. Edit the file to modify the category properties.

3. Import the modified taxonomy back into the Knowledge Tree:

   ```
   ktmgr -kt testkt -importtax testkt.tax
   ```

4. Recategorize all documents in the Knowledge Tree:

   ```
   ktmgr -kt testkt -recategorize full
   ```

   Alternative method of recategorizing all documents:

   ```
   ktmgr -kt testkt -purge -populate full
   ```

Modify category hierarchy and change some category names

1. Export the taxonomy, edit it, and import it:

   ```
   ktmgr -kt testkt -exporttax testkt.tax
   ```

2. Edit `testkt.tax`:

   ```
   ktmgr -kt testkt -importtax testkt.tax
   ```

3. Reindex the Knowledge Tree to reflect the new category hierarchy:

   ```
   ktmgr -kt testkt -reindex
   ```

4. Rebuild the index for category names and descriptions:

   ```
   ktmgr -kt testkt -indextax
   ```

Fully optimize the category index

Do a full optimization on the category index in the Knowledge Tree database:

```
ktmgr -kt d:\kts\mykt -optimize all
```

Taxonomy Management Tool (taxmgr)

The taxonomy management tool, `taxmgr`, is used to create and maintain taxonomy databases from the command line or in scripts. Taxonomy databases can be created and maintained as separate and independent entities, which is useful if a single taxonomy database is being shared by multiple Knowledge Trees.

If a taxonomy is local to a particular Knowledge Tree (created automatically by the Knowledge Tree when no external taxonomy is specified) then that taxonomy can be conveniently managed using the `ktmgr` tool.

taxmgr Syntax

The `taxmgr` syntax is:

```
taxmgr -taxonomy path [argument...]
```

Where `argument` can be any of the options listed in Table 10-6.

Table 10-6 taxmgr Arguments

Argument	Description
-taxonomy *path*	Specifies the path to the taxonomy database directory (required).
-charmap path	If the TAX file is in a character set not default for the locale, you must specify -charmap. -charmap can also be used to export TAX files in a specific character set. Specifies the character map used for importing and exporting TAX files.
-create	Creates a new taxonomy database at the specified path. Must specify style path when creating a new taxonomy database.
-debug *options*	Requests debugging output. Specify vdk, tax, or both to get debug-level output from the Verity engine.
-export *file*	Exports the categories in the taxonomy to the specified TAX file.
-import *file*	Imports new categories into the taxonomy from the specified TAX file. The existing categories in the taxonomy are deleted before importing the new ones unless the -merge flag is specified.
-index	Builds a full-text index of the names and descriptions of categories in the taxonomy. This index is used by the navigation server to find categories whose name or description match the query.
-style *file*	Specifies the path to the style directory containing the configuration and schema files for the taxonomy. It must be specified when you create a new taxonomy. Sample style files for building Knowledge Trees are located in the following directories: <install_dir>/styles/dirstyle/ <install_dir>/styles/taxstyle/
-merge	Specifies that the taxonomy being imported using the -import option should be merged into the existing taxonomy rather than replacing it. New categories are added to the existing taxonomy. Categories that already exist are modified with any new definitions in the import file.
-vdkhome path	Specifies the Verity common directory.
-verbose *options*	Requests verbose output. Specify vdk, tax, or both to get verbose output from the Verity engine.

All Verity taxonomies are created with the Root and Unused categories by default, so ktmgr for a TAX file containing only the Root and Unused categories reports:

```
Status (Tax): 0 new categories imported
```

The actual number of categories generated by the auto-generation properties of the Root category is not known until Knowledge Tree population, so ktmgr cannot report this number when importing the TAX file. The actual number of categories generated is reported at the end of the populate step. Look for a message at the end of population that looks like:

```
Indexed <number> docs into <kt dir>/taxonomy/taxonomy.vdx/parts/
00000001
<number> is the number of categories generated.
```

taxmgr Examples

The following sections provide examples for using `taxmgr`.

Create a New Taxonomy Database and Import the Taxonomy from a File

```
taxmgr -create
-taxonomy testtax
-style taxstyle/taxrule
-import mytax.tax
```

Modify the Taxonomy

1. Export the taxonomy to a file:

   ```
   taxmgr -taxonomy testtax -export testtax.tax
   ```

2. Edit the file to modify the taxonomy.
3. Import the modified taxonomy back into the taxonomy database:

   ```
   taxmgr -taxonomy testtax -import testtax.tax
   ```

To use the modified taxonomy, you must reindex and publish the Knowledge Tree.

Topic Set Conversion Tool (top2tax)

The `top2tax` tool converts a Verity knowledge base (topic set) to a taxonomy TAX file. Each named topic in the knowledge base is mapped to a category of the same name, and the tree structure of the knowledge base is mapped to the category hierarchy. For each lowest level category, the name of the corresponding Verity topic is used as the classification rule `ClassifyRule` for the category.

Using a knowledge base to construct the taxonomy for a Knowledge Tree and then populating that Knowledge Tree from a collection is a fast way of organizing the collection. However, some amount of post-processing of the taxonomy might be required in order to obtain the best results. For example, not all of the topics in the knowledge base might be appropriate as categories. If the knowledge base wasn't designed with hierarchical navigation in mind, you might need to rearrange or delete some of the categories in order to produce a logical navigation system.

It is important to choose specific classification thresholds for each of the categories in order to limit the number of categories assigned to each document. The `top2tax` tool will set a default "`ClassifyThreshold`" for each category based on the `threshold` argument to the program, but you might need to adjust the thresholds on a category by category basis in order to fine-tune the categorization.

If topics contain many evidence terms, it is likely that each document will match many category topics, though most only weakly. The classification thresholds should be set high enough that only those categories with "significant" scores are assigned to the document. A default threshold of 90 has been found to work well in many cases as an initial global setting, but results for each category should be examined closely and the individual thresholds adjusted up or down as appropriate.

Note Knowledge Trees built using a taxonomy file created by the `top2tax` tool must include the path specification to the knowledge base. If it is not included, error messages are generated, and the Knowledge Tree is not populated. When you omit the path to the topic set, you eliminate the link to the contents of the topic set.

top2tax Syntax

The following is a sample command sequence:

```
top2tax -kb <topicset_path> -taxfile example.tax
ktmgr -kt <ktpath> -collection <collpath> -create -style dirrule
ktmgr -kt <ktpath> -importtax example.tax
ktmgr -kt <ktpath> -populate full -topicset <topicset_path>
```

The syntax and a description of each argument for `top2tax` is provided in Table 10-7.

```
top2tax -kb <kb-path> -taxfile <tax-file> -rootname <rootname>
-threshold <threshold> -depth <depth> -allrules
```

Table 10-7 top2tax Arguments

Argument	Description
-allrules	Set classification rules for categories at every level in the taxonomy, not just the categories at the leaves.
-depth	The number of levels to traverse the topic set. [unlimited]
-kb	Path to the knowledge base (topic set) to convert.
-rootname	Name of the root category
-taxfile	The name of the `.TAX` file that will receive the output.
-threshold	Default classification threshold (0–100) for all categories [90].

top2tax Example

Convert “mykb.kbm” to “mytax.tax” with a default threshold of 90.

```
top2tax -kb mykb.kbm
-taxfile mytax.tax
-rootname MyData
-threshold 90
```

PART IV

Deploying

Part IV discusses how to deploy the knowledge trees you have built with Intelligent Classifier to your end users so that they can browse the categories you have created.

This section contains the following information:

- Testing Knowledge Trees and Parametric Indexes
- Deploying Knowledge Trees, Parametric Indexes, and Topic Sets
- Converting Knowledge Trees

11

Testing Knowledge Trees and Parametric Indexes

This chapter describes how to test categories, knowledge trees, and parametric indexes. The following topics are covered:

Testing from Intelligent Classifier

The following sections discuss how to test categories, taxonomies, and parametric indexes using Verity Intelligent Classifier.

Retrieving Uncategorized Documents

Not all documents are assigned to a category in the taxonomy. If you have generated a knowledge tree for the taxonomy, you can obtain a list of uncategorized documents in that taxonomy by selecting **Taxonomy | Return Uncategorized Documents**. The resulting list displays any uncategorized documents in the search results pane.

This feature is enabled only when a knowledge tree is created. When the taxonomy, topic set, or collections are changed after the knowledge tree is created, Intelligent Classifier detects the change and prompts you to regenerate the knowledge tree to retrieve uncategorized documents.

Retrieving Categories

To retrieve the category or score list that a document belongs to:

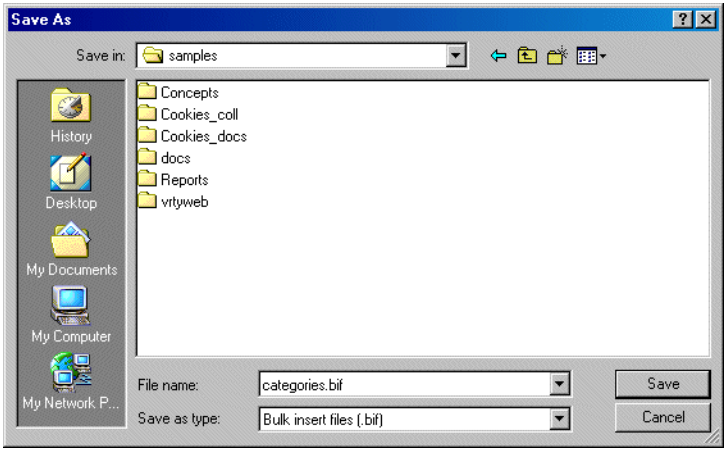
1. In the **Search Results** pane, select the document and **double-click**.
2. In the **Document View** pane, tab to the **Categories** pane.

   A score and category list is shown for the document.

   You can export your document categories into a bulk insert (.bif) file.

3. To export your categories, select **Documents | Export Document Categories**.

   This brings up the Save As dialog.



4. Enter the filename in the **File name** edit box, or select the file from the file list pane.
5. Click **Save**.

   Document category assignments for all selected documents in the results pane are written to the .bif file in the format used by `ktmgr`.

For more information about BIF files, see "BIF Files."

This feature is enabled only when a knowledge tree is created. When the taxonomy, topic set, or collections are changed after the knowledge tree is created, Intelligent Classifier detects the change and prompts you to regenerate the knowledge tree to retrieve a categories list for documents.

Testing a Category

To test a category select the category, then select **Taxonomy | Test Category**. Documents are displayed in the Search Results pane. When a category is tested, the list of documents returned includes:

- Documents from any rule or topic associated with the category
- Any manually assigned documents
- Any documents put there by automatic generation by path or metadata

Testing from the Command Line

The following sections discuss how to test categories, taxonomies, and parametric indexes from a command line.

Taxonomy and Parametric Index Search and Browse Tool (rcodk)

The `rcodk` tool is a command-line tool for searching and browsing (for documents) in a taxonomy or parametric index.

rcodk Syntax

The command-line syntax for the `rcodk` tool is:

```
rcodk [<option>]
```

rcodk Syntax Elements

Table 11-1 lists and describes the elements of the `rcodk` tool.

Table 11-1 rcodk Elements

Element	Description
`-charmap` *`charmap`*	Specifies the character map that will be used. Default is the default charmap of the specified locale.
`-collPath` *`collection_directory`*	Specifies the path to the collection that contains the source documents.
`-echo`	Instructs `rcodk` to echo the commands entered.
`-locale` *`locale`*	Specifies the locale that will be used.
`-piPath` *`pi_directory`*	Specifies the path to the parametric index directory.
`-vdkhome` *`vdkhome_directory`*	Specifies the Verity common directory.

rcodk Command Options

Table 11-2 describes the `rcodk` command options.

Table 11-2 rcodk Options

Short Command Option	Long Command	Description
`ti [<show categories flag>]`	`taxInfo`	Shows current taxonomy info, optionally show all categories.
`pi`	`piInfo`	Shows current parametric index info
`o <tax_name>`	`taxOpen`	Opens the specified taxonomy
`e <xml file> [<start cat ID>]`	`xmlExport`	Exports XML from current taxonomy to the specified XML file, beginning with the specified starting category ID (if provided).
`p [<start cat ID>]`	`print`	Prints taxonomy in XML format to the screen. If the start category ID is provided, print the sub-taxonomy starting with this category.
`g <cat ID>`	`getProps`	Shows category properties of the specified category.
`gc <cat ID> [<recursive flag>] [<max cats>] [<depth>]`	`getChildren`	Shows the specified category child categories. You can use the optional arguments to scope the operation.
`gp [<cat ID>] [<primary parent only flag>] [<recursive flag>]`	`getParent`	Shows parent categories of the specified category. When `<recursive flag>` is 1, all the ancestors of the specified category will be obtained.
`f [<ID/Name>] [<ID flag>] [<Root flag>] [<Unused flag>]`	`find`	Finds categories by ID or by name, or the Root category if `<Root_flag>` is 1, or the Unused category if `Unused_flag` is 1.
`c`	`taxClose`	Closes the current taxonomy.

Table 11-2 rcodk Options (continued)

Short Command Option	Long Command	Description
`sl [<bucket set name>] [<query] [<enum>] [<depth>] [<strict flag>]`	`select`	Specifies the bucket set selection, or list selection, without argument. `<bucket set name>` - Name of an existing bucket set in the parametric index. Use the `pi` command to find out what is in the parametric index. `[<query>]` - Bucket set query for search specific categories. For query discussion, see below. `[<enum>]` - Bucket set search by specifying `enum`. `[<depth>]` - Scope the search to the specified depth with respect to the specified buckets or categories. `[<strict Flag>]` - When `<Strict flag>` is 1, the hit count for a category or bucket only includes its own documents. Otherwise, it includes itself as well as all the descendants. Typing `sl` without any argument returns all the previous selections.
`cl`	`clear`	Clears all bucket set selections.
`s [<collection query>]`	`search`	Searches using the parametric index.
`r [<starting result index>] [<print category enums flag>]`	`retrieve`	Retrieves and print search results.
`df [<field name><field width>...]`	`fields`	Lists of fields to display during search result retrieval. `<field name>` and `<field width>` must be in pair.
`b [<starting result index>]`	`browse`	Browses results of last parametric index search starting from the specified result index.
`u`	`user`	Prompts for user authentication credentials.
`q`	`quit`	Leaves the application.
`?`	`help`	Displays help text.

Query Operations

Use Parametric Index Structured Query Language (PI SQL) statements with the `sl` command. Query operations `s` and `b` depend on the buckets you select. You can refine the selection for a taxonomy by specifying categories in the category hierarchy using PI SQL statements to focus the selection. Query operations are performed in the following way:

1. Buckets are selected from a bucket set using the algorithm.
2. Free-text search criteria are applied.
3. When the bucket set holds a taxonomy, the category depth is used to limit selections from the selected buckets.
4. When the bucket set holds a taxonomy, a PI SQL statement further refines the selection.

The syntax of a PI SQL statement is as follows:

`[ xml_path operator ] operand`

where *`xml_path`* is an optional path to a tag in the taxonomy definition, *`operand`* is a valid PI SQL statement, and *`operator`* is one of the operators described in Table 11-3.

Table 11-3 Query Operators

Operator	Description
`<phrase>`	Match the exact phrase
`<contains>`	Match the specified words in any order
`<and>`	Intersect (logical `and`)
`<or>`	Union (logical `or`)
`<not>`	Negate (logical `not`)
`<, <=, >, =>`	Less than, less than or equal, greater than, greater than or equal, respectively, to the specified value
`=`	Equal to a value that may contain the following wildcard operators: * —match any characters ? —match a single character

The *`xml_path`* is the constant `Category`, which specifies any category, including reference categories and the root category, or a path to a relative (floating) attribute. You separate tags in the path with colons (:). Attribute names are prefixed with a dollar sign ($). For example, you can specify all categories with names containing the word `VRTY` as follows: `Category:$Name <CONTAINS> VRTY`.

A shortcut for specifying a category by name or ID is provided.

```
sl <bucket set name> category_name
```

is equivalent to

```
sl <bucket set name> Category:$Name=category_name
```

Similarly,

```
s1 <bucket set name> #category_id
```

is equivalent to

```
s1 <bucket set name> Category:$CategoryID=category_id
```

`_Uncategorized` is a reserved Category Name and Category ID specifically for retrieving uncategorized documents in a bucket set. To do this, type:

```
sl <bucket set name> #_Uncategorized
```

or

```
s1 <bucket set name> _Uncategorized
```

Note For complete information about PI SQL capabilities, see the *Verity K2 Parametric Developer Guide*.

Examples

The following sample command uses `mkpi` to create a new parametric index called `cookies.pi` with the associated collection `Cookies_coll` (both from the Verity samples that ship with K2 Enterprise).

```
mkpi -piPath cookies.pi -create -collPath "..\..\..\..\
Intelligent Classifier\samples\Cookies_coll"
-outline cookies_pi.xml
MKPI - Verity, Inc. Version 5.0.0 (Build 20030403)
Creating new pi: "cookies.pi" from outline file:
"cookies_pi.xml"
BucketSet: Cities
BucketSet: Music
BucketSet: Path
BucketSet: Software
BucketSet: TV
TaxDef: Cookies
Indexing from collection: ..\..\..\..\Intelligent Classifier\
```

```
samples\Cookies_coll
Indexed 3475 documents to 00000001.pi
Successfully indexed a total of 3475 documents
mkpi done
```

The following `rcodk` example starts `rcodk` to search and browse the specified parametric index called `cookies.pi`, which it associates with `Cookies_coll`.

```
C:\Program Files\Verity\IC>rcodk
-piPath "C:\Program Files\Verity\Data\ktrees\cookies\IC\
cookies.pi"
-collPath "C:\Program Files\Verity\Intelligent Classifier\
samples\Cookies_coll"
rcodk - Verity, Inc. Version 5.0.0 (_nti40, Apr  9 2003)
Type 'help' for a list of commands.
Reading in PI: "C:\Program Files\Verity\Data\ktrees\cookiesIC\
cookies.pi"
```

You can use the short version of a command (for example, `pi`) or the long version (for example, `piInfo`). The long version is useful for command scripts. To view the available commands, type:

```
rcodk> ?
```

The following command shows the current parametric index information.

```
rcodk> pi
Information for pi: C:\Program Files\Verity\Data\ktrees\cookies\
IC\cookies.pi
Alias: C:\Program Files\Verity\Data\ktrees\cookies\IC\cookies.pi
Outline:
Description:

Number of BucketSetDefs:    5
  1:  name="Cities"  type="Taxonomy"  taxDef="Cookies"
startNode="Root/Cities"
  2:  name="Music"  type="Taxonomy"  taxDef="Cookies"
startNode="Root/Music"
  3:  name="Path"  type="Taxonomy"  taxDef="Cookies"
startNode="Root/Path"
  4:  name="Software"  type="Taxonomy"  taxDef="Cookies"
startNode="Root/Software"
  5:  name="TV"  type="Taxonomy"  taxDef="Cookies"
startNode="Root/TV"
```

```
Number of TaxDefs:          1
  1:  name="Cookies"  xmlFile="cookies_tax.xml"
Description="Taxonomy for sample collection Cookies_coll"
lastUpdate="2003-04-09 14:04:42"

Number of TopicSetDefs:     0

Number of VdkFieldDefs:     0

Number of KTreeDefs:        5
  1:  name="Cities"  tax="Cookies"  startCat="Root/Cities"
  2:  name="Music"  tax="Cookies"  startCat="Root/Music"
  3:  name="Path"  tax="Cookies"  startCat="Root/Path"
  4:  name="Software"  tax="Cookies"  startCat="Root/Software"
  5:  name="TV"  tax="Cookies"  startCat="Root/TV"
rcodk> o Cookies
rcodk> ti 1
Information for taxonomy: Cookies
Definition file path: cookies_tax.xml
Description: DESCRIPTION_IN_OUTLINE: Taxonomy for sample
collection Cookies_coll
            DESCRIPTION_IN_TAXONOMY: Taxonomy using query rules
and category generation by path

Categories:
ID: Root  Name: Root
ID: Music  Name: Music
ID: Elvis  Name: Elvis
ID: Rolling_Stones  Name: Rolling Stones
ID: Beatles  Name: Beatles
ID: Bruce_Springsteen  Name: Bruce Springsteen
ID: Bob_Dylan  Name: Bob Dylan
ID: Cities  Name: Cities
ID: New_York  Name: New York
ID: San_Francisco  Name: San Francisco
ID: Los_Angeles  Name: Los Angeles
ID: Software  Name: Software
ID: Programmer  Name: Programmer
ID: Hacker  Name: Hacker
ID: TV  Name: TV
ID: Star_Trek  Name: Star Trek
ID: Cheers  Name: Cheers
ID: Path  Name: Path
ID: Path_zippy  Name: zippy
```

```
ID: Path_work  Name: work
ID: Path_wisdom  Name: wisdom
ID: Path_startrek  Name: startrek
ID: Path_sports  Name: sports
ID: Path_songs-poems  Name: songs-poems
ID: Path_science  Name: science
ID: Path_riddles  Name: riddles
ID: Path_politics  Name: politics
ID: Path_platitudes  Name: platitudes
ID: Path_people  Name: people
ID: Path_news  Name: news
ID: Path_miscellaneous  Name: miscellaneous
ID: Path_men-women  Name: men-women
ID: Path_medicine  Name: medicine
ID: Path_magic  Name: magic
ID: Path_love  Name: love
ID: Path_literature  Name: literature
ID: Path_linuxcookie  Name: linuxcookie
ID: Path_law  Name: law
ID: Path_kids  Name: kids
ID: Path_humorists  Name: humorists
ID: Path_fortunes  Name: fortunes
ID: Path_food  Name: food
ID: Path_ethnic  Name: ethnic
ID: Path_education  Name: education
ID: Path_drugs  Name: drugs
ID: Path_definitions  Name: definitions
ID: Path_cookie  Name: cookie
ID: Path_computers  Name: computers
ID: Path_art  Name: art
ID: Unused  Name: Unused Categories
```

The `f` command in the following example, takes a category ID or name and prints the ID and name only:

```
rcodk> f Path
ID: Path  Name: Path
```

The `g` command takes a category ID only and prints all category properties:

```
rcodk> g Path
Properties for Category ID: Path
Name: Path
Description:
```

```
Classify Rule:
Threshold: 0
Auto-categorization method: Path
Auto-categorization field: DOC_FN
Auto-categorization category delimiter:
Auto-categorization hierarchy delimiter: /
Auto-categorization ignore prefix: Cookies_docs/
Reference category: No
Refining category: No
Sort child categories: Yes
Concept Words:
User defined properties:
```

In the following example, the `gc` command prints immediate children only by default:

```
rcodk> gc Root
ID: Music  Name: Music
ID: Cities  Name: Cities
ID: Software  Name: Software
ID: TV  Name: TV
ID: Path  Name: Path
```

In the following example, the `gc` command recursively prints all children with recursive flag set to 1.

```
rcodk> gc Root 1
ID: Music  Name: Music
ID: Elvis  Name: Elvis
ID: Rolling_Stones  Name: Rolling Stones
ID: Beatles  Name: Beatles
ID: Bruce_Springsteen  Name: Bruce Springsteen
ID: Bob_Dylan  Name: Bob Dylan
ID: Cities  Name: Cities
ID: New_York  Name: New York
ID: San_Francisco  Name: San Francisco
ID: Los_Angeles  Name: Los Angeles
ID: Software  Name: Software
ID: Programmer  Name: Programmer
ID: Hacker  Name: Hacker
ID: TV  Name: TV
ID: Star_Trek  Name: Star Trek
ID: Cheers  Name: Cheers
ID: Path  Name: Path
ID: Path_zippy  Name: zippy
```

```
ID: Path_work  Name: work
ID: Path_wisdom  Name: wisdom
ID: Path_startrek  Name: startrek
ID: Path_sports  Name: sports
ID: Path_songs-poems  Name: songs-poems
ID: Path_science  Name: science
ID: Path_riddles  Name: riddles
ID: Path_politics  Name: politics
ID: Path_platitudes  Name: platitudes
ID: Path_people  Name: people
ID: Path_news  Name: news
ID: Path_miscellaneous  Name: miscellaneous
ID: Path_men-women  Name: men-women
ID: Path_medicine  Name: medicine
ID: Path_magic  Name: magic
ID: Path_love  Name: love
ID: Path_literature  Name: literature
ID: Path_linuxcookie  Name: linuxcookie
ID: Path_law  Name: law
ID: Path_kids  Name: kids
ID: Path_humorists  Name: humorists
ID: Path_fortunes  Name: fortunes
ID: Path_food  Name: food
ID: Path_ethnic  Name: ethnic
ID: Path_education  Name: education
ID: Path_drugs  Name: drugs
ID: Path_definitions  Name: definitions
ID: Path_cookie  Name: cookie
ID: Path_computers  Name: computers
ID: Path_art  Name: art
```

In the following example, the `gp` command recursively prints all parents up to the root with the `recursive` flag set to 1. The `gp` command prints parent and primary parent category lists with `primary` flag set to 0. The lists are the same except for reference categories.

```
rcodk> gp Path_art 0 1
Primary parent categories:
ID: Path  Name: Path
ID: Root  Name: Root
Parent categories:
ID: Path  Name: Path
ID: Root  Name: Root
```

In the following example, this select and search combination searches the document collection using the query "Yeats" and allows the `retrieve` command to print the categories in the *`Path`* bucket set to which the returned documents are assigned.

```
rcodk> sl Path
rcodk> s Yeats
VDK: (661 ms) Elapsed Retrieval
VDK search retrieved 6/3475 documents
PI  search retrieved 6/3475 documents
```

In the following example, setting the `retrieve` command's print category `enums` flag to 1 prints category information.

```
rcodk> r 1 1
Number  SCORE  VdkVgwKey
CatScore  CatEnum
1:    0.7742 Cookies_docs/computers/segment127.txt
1.0000    Path/computers
2:    0.7742 Cookies_docs/computers/segment268.txt
1.0000    Path/computers
3:    0.7742 Cookies_docs/men-women/segment132.txt
1.0000    Path/men-women
4:    0.7742 Cookies_docs/men-women/segment67.txt
1.0000    Path/men-women
5:    0.7742 Cookies_docs/people/segment113.txt
1.0000    Path/people
6:    0.7742 Cookies_docs/people/segment221.txt
1.0000    Path/people
```

In the following example, a null collection search is performed that retrieves all documents from the collection, but the results are refined by the selections to those documents that are assigned to the art category and the *`Bob Dylan`* category

```
rcodk> cl
rcodk> sl Path art
rcodk> sl Music "Bob Dylan"
rcodk> s ""
VDK: (10 ms) Elapsed Retrieval
VDK search retrieved 3475/3475 documents
PI  search retrieved 6/3475 documents
rcodk> r 1 1
Number  SCORE  VdkVgwKey
CatScore  CatEnum
1:          Cookies_docs/art/segment94.txt
1.0000    Path/art
```

```
0.7742    Music/Bob Dylan
2:          Cookies_docs/art/segment84.txt
1.0000    Path/art

0.7742    Music/Bob Dylan
3:          Cookies_docs/art/segment66.txt
1.0000    Path/art

0.7742    Music/Bob Dylan

0.7742    Music/Beatles
4:          Cookies_docs/art/segment31.txt
1.0000    Path/art

0.7742    Music/Bob Dylan
5:          Cookies_docs/art/segment17.txt
1.0000    Path/art

0.7742    Music/Bob Dylan
6:          Cookies_docs/art/segment119.txt
1.0000    Path/art

0.7742    Music/Bob Dylan

0.7742    Music/Beatles
```

In the following example, the `sl` command with no options shows the current selections.

```
rcodk> sl
Number of selections:    2
  1:  name="Path"  queryQuestion="art" depth = "0"
showStricts="off"
  2:  name="Music"  queryQuestion="Bob Dylan" depth = "0"
showStricts="off"
rcodk> b
BucketSet Path had 6 total hits
Path(6)
Path/art(6)

BucketSet Music had 6 total hits
Music(6)
Music/Bob Dylan(6)
```

In the following example, the selections are cleared and redefined to be the same as before, but with the strict flag set to 1. The differences caused by the strict flag are illustrated by the browse results as follows:

```
rcodk> cl
rcodk> sl Music "Bob Dylan" "" 0 1
rcodk> sl Path art "" 0 1
rcodk> sl
Number of selections:    2
  1:  name="Music"  queryQuestion="Bob Dylan"  showStricts="on"
  2:  name="Path"  queryQuestion="art" depth = "0"
showStricts="on"
rcodk> s ""
VDK: (10 ms) Elapsed Retrieval
VDK search retrieved 3475/3475 documents
PI  search retrieved 6/3475 documents
rcodk> r 1 1
Number  SCORE  VdkVgwKey
CatScore  CatEnum
1:          Cookies_docs/art/segment94.txt
0.7742    Music/Bob Dylan

1.0000    Path/art
2:          Cookies_docs/art/segment84.txt
0.7742    Music/Bob Dylan

1.0000    Path/art
3:          Cookies_docs/art/segment66.txt
0.7742    Music/Bob Dylan

0.7742    Music/Beatles

1.0000    Path/art
4:          Cookies_docs/art/segment31.txt
0.7742    Music/Bob Dylan

1.0000    Path/art
5:          Cookies_docs/art/segment17.txt
0.7742    Music/Bob Dylan

1.0000    Path/art
6:          Cookies_docs/art/segment119.txt
0.7742    Music/Bob Dylan
```

```
0.7742    Music/Beatles

1.0000    Path/art
```

In the following example, the browse results show no hits for the parent categories because the strict flag is on the browse. The default behavior is to propagate hits in child categories up to the parents as well. The `strict` setting means to show hits only for categories to which documents are "strictly" assigned.

```
rcodk> b
BucketSet Music had 6 total hits
Music(0)
Music/Bob Dylan(6)

BucketSet Path had 6 total hits
Path(0)
Path/art(6)
```

Knowledge Tree Search Tool (ktsrch)

The `ktsrch` tool is a simple program that runs a *scoped search* (a scoped search is a search that is limited to specified categories) against a Knowledge Tree and displays the top 20 results grouped by category. `ktsrch` is useful for quickly verifying the validity of a Knowledge Tree.

ktsrch Syntax

The syntax for `ktsrch` is:

```
ktsrch <category>
<kt-query>
<collection-query>
<kt>
<collection>
```

Table 11-4 describes the arguments for `kctsrch`.

Table 11-4 ktsrch Arguments

Argument	Description
category	Specifies the category ID of the category in which to scope the search.
`kt`-query	Specifies the filter query applied to the Knowledge Tree. It is AND-ed with the category scope.
collection-query	Specifies the query applied to the collection within the specified scope.
`kt`	Specifies the Knowledge Tree to search.
collection	Specifies the collection to which the Knowledge Tree is attached.
-locale <locale_name>	Specifies the locale name to be used to search the Knowledge Tree (optional).
-charMap <charMap_name>	Specifies the character map name to be used to search the Knowledge Tree (optional). The default character map is 1252.

ktsrch Examples

Search for the word "query" scoped to the Root category:

```
ktsrch root "" "query" testkt testcoll
```

Search for the phrase "data sheet" scoped to the Marketing category:

```
ktsrch "Marketing" "" "data sheet" testkt testcoll
```

Optimized Knowledge Tree Search Tool (qsrch)

The `qsrch` tool is a command-line tool for searching (for documents) in an optimized Knowledge Tree. An optimized Knowledge Tree is a Knowledge Tree that has been optimized using `ktmgr` tool's `-k2` option. Using `qsrch` you can check to be sure that an optimized Knowledge Tree returns expected results before you make it available in the Verity Intelligent Classifier application.

qsrch Syntax

The syntax for `qsrch` is:

```
qsrch -kt <tree_path>
-locale <locale_name>
-charMap <charMap_name>
```

Table 11-5 describes the arguments for `qsrch`.

Table 11-5 qsrch Arguments

Argument	Description
-kt <tree_path>	Specifies the path name to the Knowledge Tree to be searched.
-locale <locale_name>	Specifies the locale name to be used to process the search (optional).
-charMap <charMap_name>	Specifies the character map name to be used to process the search (optional). The default character map is 1252.

qsrch Command Options

The `qsrch` command options are described in Table 11-6.

Table 11-6 qsrch Options

Command Option	Description
m [number]	Specifies the maximum number of results to be retrieved. For a collection search, results are documents. For a Knowledge Tree search, documents are returned. For a Knowledge Tree category search, categories are returned.
w [start] [page]	Specifies the results list window to display for Knowledge Tree category searches. ■ [start] is the starting row number in the results list. ■ [page] is the number of categories to display in the window.
r [start number]	Specifies the row number for the first item to display in the results list for Knowledge Tree document searches.
s [-strict] [cat ID] [query]	Specifies search parameters for a Knowledge Tree category search. ■ [-strict] specifies whether a category search returns stricts only. By default, a category search includes stricts and descendants. ■ [cat ID] specifies a specific category ID to be used to process a category search. ■ [query] is the Verity query to be used to process the search request.

12

Deploying Knowledge Trees, Parametric Indexes, and Topic Sets

After you have used Intelligent Classifier to create and build your knowledge tree, you need to deploy the tree so that your users can browse the categories attached to your collections. This chapter contains the following sections:

Managing Knowledge Trees

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

The first step in deploying your knowledge tree is to register it to a host and then attach it to a K2 Server. For information on how to configure a K2 server, see the *Verity K2 Dashboard Administrator Guide.* In addition, this guide also provides all the information you need to manage your knowledge trees for deployment, including:

- Editing Knowledge Tree Properties
- Configuring the Knowledge Tree
- Changing the State of a Knowledge Tree
- Assigning or Disabling Knowledge Tree Security
- Detaching a Knowledge Tree from a K2 Server
- De-registering a Knowledge Tree from a Host
- Using Expert Knowledge Tree Settings
- Testing Search on a Server

Configuring Parametric Indexes

The *Verity K2 Dashboard Administrator Guide* describes how to administer an existing parametric index on a K2 system. It includes the following sections:

- Editing Parametric Index Properties
- Changing the State of a Parametric Index
- Detaching a Parametric Index from a K2 Server
- De-registering a Parametric Index from a Host

For complete information on `mkpi`, Verity Parametric Selection, and testing a parametric index with `rck2`, see the *Verity K2 Parametric Developer Guide*.

Configuring Topic Sets in K2 Server

After you have created a topic set, you can configure it for use with Verity K2.

1. Build a topic set, as described in "Using the Topic Editor" and in the *Verity Query Language and Topic Guide*.
2. Configure the K2 server.

To configure a topic set or a knowledge tree for use with Verity K2 Enterprise Server:

1. Build a topic set.
2. Enter the **Topic Set Path** in the **Expert Settings** of the K2 Dashboard for K2 Server. For more information about configuring a K2 server, see the *Verity K2 Dashboard Administrator Guide*.

Note All topics and topic sets are recognized by the search engine as valid topics in user queries. For the best search processing performance, use the `TopicSet` parameter to specify a single topic set. You can also serve more than one topic set using knowledge bases, as described in Part III, "Populating Taxonomies."

For the best search processing performance, use the **Topic Set Path** to specify a single topic set.

1. In the K2 Dashboard K2 Server Expert Settings, for **Topic Set Path**, type the default path name to the directory that contains the topic set you want to use. The value of **Topic Set Path** identifies the default topic set to make available to clients at startup by every search service.
2. Start the K2 Server.

   The topics identified in the named **Topic Set Path** will now override words used in searching.

You can serve multiple topic sets using knowledge bases. In the **Knowledge Base Path**, type the path name to a knowledge base map file (see "Using Configuration Files" for information about this file). The value of **Knowledge Base Path** identifies the topic sets (multiple) to make available to clients at startup.

13

Converting Knowledge Trees

This chapter helps you determine whether you need to modify your existing knowledge trees.

- You must recreate your knowledge tree if the knowledge tree was based on a collection that was recreated for V6.1

  See *Verity K2 Migration Guide* for information on which collections require recreating for K2 6.1 and for the default locale in each release.

Note Existing scripts that create, maintain, or update knowledge trees might need to be modified to account for a change in the default locale. If the scripts do not specify a locale, then the new default locale is used. To use one of the previous default locales, update the scripts to specify the locale with the `-locale` command-line option .

This chapter includes the following sections:

Converting Knowledge Trees to Trees in Parametric Indexes

You can convert your knowledge trees to trees in parametric indexes using Verity Intelligent Classifier. This allows you to take advantage of the powerful relational taxonomies in parametric indexes.

Note If you convert your knowledge trees to knowledge trees in parametric indexes, you must change your knowledge tree application to a parametric application. See the *Verity K2 Parametric Developer Guide* and the *Verity Organization Developer's Kit Programming Guide* for more information.

To create a parametric index from an existing knowledge tree, Intelligent Classifier must be able to access the knowledge tree's workspace stored in the `IC` directory. If the knowledge tree was originally created in Intelligent Classifier, the `IC` directory will be available (see "Converting Knowledge Trees Created by Intelligent Classifier" on page 244). If the knowledge tree was created by the knowledge tree management command-line tool, `ktmgr`, the `IC` directory might not exist. In this case, the `IC` directory must be created manually before you can create the parametric index (see "Converting Knowledge Trees Created by ktmgr" on page 248).

Converting Knowledge Trees Created by Intelligent Classifier

If the knowledge tree was originally created by Intelligent Classifier, or was created by `ktmgr` and has an `IC` directory, follow these steps:

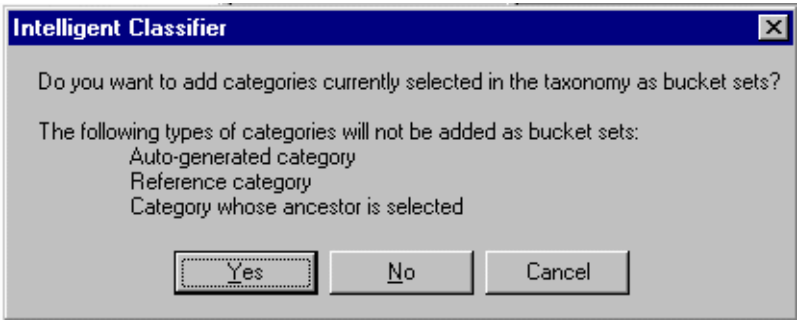
1. Open the knowledge tree in Intelligent Classifier using **File** | **Open Knowledge Tree**.
2. If there is more than one collection in the workspace, ensure only one is loaded. Press **F2**, or select **File** | **Add/Remove Collections**. In the Collections dialog, ensure only one collection is listed.

   When you create a parametric index there can be only one collection in the workspace. You must generate a separate parametric index for each collection in the knowledge

tree. To replicate the behavior of the original knowledge tree, you must include all the parametric indexes in your search. See "Replicating the Behavior of the Original Knowledge Tree" on page 247 for more information.

3. Select the **Root** node of the taxonomy. This node name is converted to a bucket set of the same name.
4. Choose **File | Create Parametric Index**. You are prompted to add the selected category as a bucket set.

Figure 13-1 Add Categories dialog

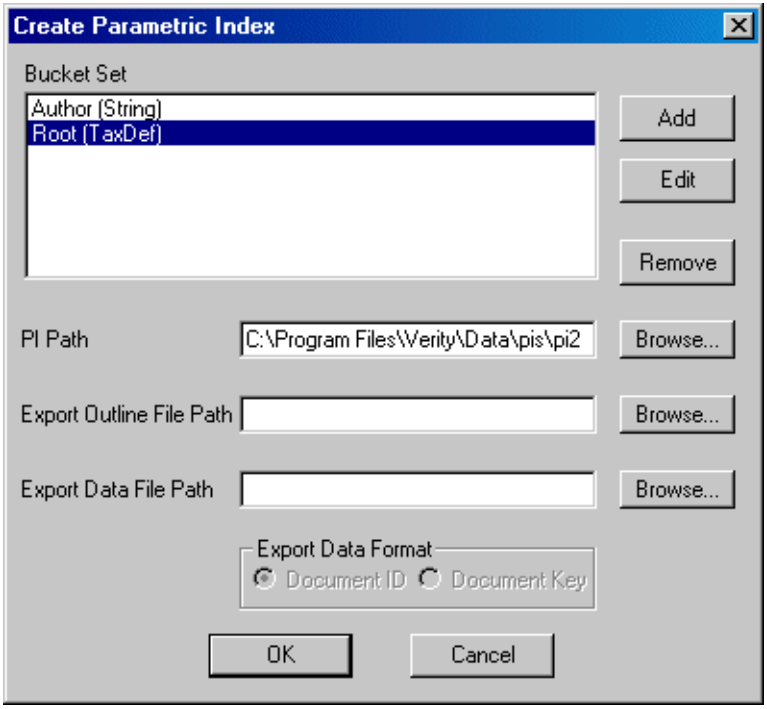


5. Click **Yes** to add the selected category as a bucket set in the parametric index. The Root node is converted to a bucket set, and all its children are converted to buckets within that bucket set.

Note If the knowledge tree has multiple collections, each parametric index created from that knowledge tree must have the same bucket set name. This enables you to replicate the behavior of the original knowledge tree by merging the parametric indexes at runtime.

The **Create Parametric Index** dialog appears.

Figure 13-2 Create Parametric dialog



The text in the parentheses following each bucket set name indicates its type. **TaxDef** indicates the bucket set is based on a category in the current taxonomy.

6. In the **PI Path** field, specify the location and name of the parametric index. Click **OK**.

   A dialog appears stating that you have successfully created a parametric index.

7. Create a parametric index, following the steps above, for each collection in the knowledge tree.

After all the parametric indexes are created, add them to the K2 system using K2 Dashboard or the command-line tool `rcadmin`. You must:

- register the index with a host machine
- attach the index to a K2 Server
- perform a quick restart on the K2 Server

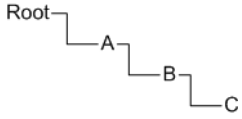
See the *Verity K2 Dashboard Administrator Guide* or the *Verity K2 rcadmin Guide* for more information.

Replicating the Behavior of the Original Knowledge Tree

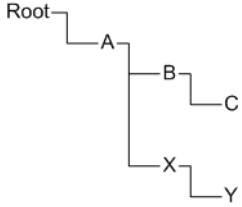
To replicate the behavior of the original knowledge tree, a search must include *all* the parametric indexes generated from the knowledge tree.

For example, if the original knowledge tree has two collections: `c1` and `c2`, and you create a parametric index (`pi1`) based on `c1`, and another parametric index (`pi2`) based on `c2`. The bucket set is named `BS1`.

The tree structure for bucket set `BS1` in `pi1` is:



The tree structure of bucket set `BS1` in `pi2` is:



To replicate browsing the category node `C` in the original knowledge tree, you would use the following commands in `rck2`:

```
rck2> p pi1 pi2
rck2-p> bucket Root/A/B/C:BS1
rck2-p> s
s
Elapsed time:30(ms)
numHits = 1796, numProc = 12940
rck2-p>
```

Documents are returned from node `C` in `pi1` *and* `pi2`, respectively.

Converting Knowledge Trees Created by ktmgr

Knowledge trees that were created using `ktmgr` may not have an `IC` directory. In order to convert these knowledge trees to trees in parametric indexes using Intelligent Classifier, an `IC` directory based on the knowledge tree must first be created.

If the knowledge tree was created using `ktmgr`, follow these steps:

1. In Intelligent Classifier, create a new knowledge tree using the Classification Wizard. Select **File** | **Classification Wizard**.
2. Enter the name and location of the new knowledge tree. Click **Next**.
3. Add the collection used in the knowledge tree created by `ktmgr`. Click **Next**.

   If the knowledge tree has only one collection, the `PrimaryPath` field in the `dir.cfg` file specifies the path to that collection. If the knowledge tree has multiple collections, the `PrimaryPath` field in the `dir.cfg` file specifies a path to a `clm` file that contains the paths to all collections in the knowledge tree.
4. In the Automatic Classification dialog, select **None**. Click **Next**, and then click **Finish**.
5. Import the topic set(s) for the knowledge tree.
   a. Close the default topic set. Select **Topic** | **Close Current Topic Set**.
   b. Select **Topic** | **Open Topic Set** | **Topic Set**. If the knowledge tree has more than one topic set, open each topic set separately.

      If the knowledge tree has only one topic set, the `KBPath` field in the `dir.cfg` file specifies the path to that topic set. If the knowledge tree has multiple topic sets, the `KBPath` field in the `dir.cfg` file specifies a path to a `kbm` file that contains the paths to all topic sets in the knowledge tree.
6. Close the knowledge tree workspace.
7. Using `ktmgr`, export the taxonomy from the original knowledge tree, and save the taxonomy as *`newkt_dir`*`\IC\workspace.tax`, where *`newkt_dir`* is the location of the new knowledge tree you created in Intelligent Classifier:

   ```
   ktmgr -kt kt_dir -exporttax newkt_dir\IC\workspace.tax
   ```

   where *`kt_dir`* is the location of the original knowledge tree, and *`newkt_dir`* is the location of the new knowledge tree.

8. If there are no documents assigned manually in the knowledge tree, go to step 9. Otherwise, use `extract.exe` to generate a bulk insert file (BIF), which stores the manual assignments, and save the BIF as *`newkt_dir`*`\IC\workspace.bif`:

```
extract -locale localename -n -credentials user_id newkt_dir\IC\
workspace kt_dir\index\category.vdx


rename newkt_dir\IC\workspace.vdk newkt_dir\IC\workspace.bif
```

   where *`kt_dir`* is the location of the original knowledge tree, and *`newkt_dir`* is the location of the new knowledge tree.

   If a Verity locale is not specified, the default is used.

   The `IC` folder has now been created.

9. To generate the parametric index, follow the steps in "Converting Knowledge Trees Created by Intelligent Classifier" on page 244.

APPENDIXES

A

Setting Style Options

The behavior of some aspects of Intelligent Classifier is affected by the options used when a collection is built. This appendix contains the following sections:

style.prm Options

When a collection is built, the `style.prm` file (in the collection's `style` directory) configures certain aspects of the collection's index.

Note The `style.prm` file in Intelligent Classifier's installation directory is for Intelligent Classifier's internal use only.

- By default, a *Soundex index* is not built for a collection, so `<Soundex>` will not work unless a line such as:

  ```
  $define WORD-IDXOPTS "Soundex"
  ```

  is defined in `style.prm`. (You can also have other options, like "Stemdex" and "Casedex", on that line, too.)

- By default, `<Paragraph>` and `<Sentence>` match documents if the search terms occur within a certain distance of each other, whether or not the search terms occur in the same paragraph or sentence. This is controlled by the line

  ```
  $define IDX-CONFIG "WCT"
  ```

  To enable `<Paragraph>` and `<Sentence>` to match documents only when the search terms occur in the same paragraph or sentence, change this to

  ```
  $define IDX-CONFIG "PSW"
  ```

  or

  ```
  $define IDX-CONFIG "PSW Many"
  ```

- Adding "Many" affects the behavior of the `<Many>` modifier. Without it, `<Many>` scores documents solely according to the number of occurrences of the search term. With it, `<Many>` also takes the length of the document into account and scores documents according to the ratio of occurrences to the document length.

- If the PSW option is used, then `<Near>` and `<Near/n>` will not cross sentence or paragraph boundaries. That is, they will only return documents where the search terms are within *n* words *and* are in the same sentence and paragraph.

  - If PSW is used, the recommended operators to use are `<Paragraph>` and `<Sentence>`.

  - If WCT is used, the recommended operators to use are `<Near>` and `<Near/n>`.

- The document cluster extraction feature and automatic categorization based on document clusters or on manually assigned documents will not work unless the *document features vector* is defined. This can be done by including the following line in the `style.prm` file.

  ```
  $define DOC-FEATURES "TF"
  ```

style.lex options

- The `style.lex` file affects Intelligent Classifier in the following ways:
  - It determines how non-alphanumeric characters (such as &, /, and ") are treated. For example, the default `style.lex` settings do not include apostrophes as part of a word, but treat it as punctuation and ignore it. So a word like "computer's" is indexed as two words: "computer" followed by "s".
  - It can also be configured to exclude words below a certain minimum length.
- The EOS setting in `style.lex` also affects what punctuation marks the end of sentences, which affects the `<Sentence>` operator if the PSW option is used.

You do not usually have to modify the `style.lex` file. The AutoPhrase feature is provided to avoid the need to know what lexer is being used.

Fields and Zones

The `<Field>` and `<In>` operators are useful if the collection has zones and fields defined in it.

Assists Window

The Assists window has the following restrictions.

- The Assists window (see "Using the Assists Dialog") only shows the `<Stem>`, `<Wildcard>`, and `<Word>` assists if the collection has *word list assist* enabled. And it only shows the `<Typo/2>` assist if the collection has the *ngram index* added. To build both of these, use `mkvdk` with the following argument:

  ```
  -optimize spanword-ngramindex
  ```

  Note Do not also use the `-words` argument.

- The Assists window only shows the Zone and Field assists if the collection has zones and fields defined in it.

B

Using Taxonomy Definition Files

Taxonomies are persisted in two types of file formats. From Intelligent Classifier you can import and export a taxonomy in both of these formats.

TAX Files

Taxonomy files in TAX format use `.tax` as the filename extension. A category node is described by a block of property name-value string pairs. An empty line ends the description of a category node. A property-value pair must be on one line, starting with a property name, a colon and space, and the value of the property. For example:

```
Name: Root
```

If the value of a property starts or ends with spaces, the value string must be enclosed in quotes. For example:

```
Name: "Document Types"
```

Basic Properties

You will commonly see the properties listed in Table B-1 in a category description block.

Table B-1 Basic Properties

Property	Description
`category`	The ID of a category. The ID of a primary category, not a referenced category, must be unique in a taxonomy. The ID of a referenced category must be the same as the category to which it refers.
`Name`	The name of a category. A category name should be user-friendly.
`primary-parent`	The ID of the parent of the current category. Specifies that the current node is a primary category.
`parent`	The ID of the parent of the current category. Specifies that the current node is a referenced category.
`ClassifyRule`	The rule associated with the current category. If the rule is a topic, the value string must be formatted as {topicSetName:topicName}.
`ClassifyThreshold`	The score threshold of the rule. Value can be from 0 to 99.

Auto-categorization Properties

Table B-2 describes category properties that are used for dynamic auto- categorization.

Table B-2 Auto-categorization Properties

Property	Description
`AutoCatField: field`	*field* is the field from which categories are to be generated, for example, *Author*. These fields must be in a collection's style file.
`AutoCatExtract: file_or_url`	*file_or_url* is either `File` or `URL` and performs the same function as the `/extract` field modifier.
`AutoCatPrefixIgnore: prefix`	You can use *prefix* to cause this prefix to be ignored from the `AutoCatField` prior to auto categorization.
`AutoCatCatDelim: delim`	*delim* is the field delimiter. The whole string is used to delimit category IDs in the field specified by `AutoCatField`.
`AutoCatHierDelim: sep`	*sep* is the hierarchy or path separator for the field.

Additional Properties

Table B-3 describes additional properties that can be used for category nodes.

Table B-3 Additional Properties

Property	Description
`RefiningCategory: ref`	*ref* is 0 or 1. If this property is set to 1, this category is a refining category, so the associated rule is only applied against the documents that match this category's parent's rule.
`SortChildProp`	Specifies the property used for sorting the child nodes of the current node. If absent, `Name` is used.
`Description`	Provides more description for the current category.
`Created`	Specifies the time the current node was created.
`Modified`	Specifies the time when the current node was modified.

In addition to the standard properties, you can also define your own *user-defined properties* for a category node.

Note on Topic Set Names

Intelligent Classifier uses an underbar in topic set names rather than spaces, so "a b c" becomes "a_b_c". A taxonomy created with an earlier version could refer to a topic set name that still has spaces. If you import such a taxonomy into Intelligent Classifier, the taxonomy cannot find its topic set.

If you have taxonomies that refer to topic sets with spaces in their names, you need to edit your TAX file to replace the spaces with underbars. For example, before editing the following clip from an old TAX file where "a b c" is the name of the topic set:

```
category: Root
Name: Root

category: Document_Types
Name: Document_Types
ClassifyRule: {a b c:Document_Types}
primary-parent: Root

category: As_Built
Name: As_Built
ClassifyRule: {a b c:As_Built}
primary-parent: Document_Types
```

You would need to edit the previous example by replacing the string “a b c” with the string “a_b_c”. After editing it would look as follows:

```
category: Root
Name: Root
Created: 30-Mar-2002 02:14:03 pm
Modified: 30-Mar-2002 02:14:03 pm

category: Document_Types
Name: Document_Types
ClassifyRule: {a_b_c:Document_Types}
primary-parent: Root

category: As_Built
Name: As_Built
ClassifyRule: {a_b_c:As_Built}
primary-parent: Document_Types
```

XML Taxonomy Files

Taxonomy files can be imported and exported in XML format. A DTD file defines the XML format for taxonomy representation. This DTD file (taxonomy.dtd) is located in `IC_install\common\`, where `IC_install` is the directory where you installed Intelligent Classifier.

The following is the content of `taxonomy.dtd`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- A taxonomy consists of a root category with a fixed CategoryID
"Root" -->
<!-- and an unused category with a fixed CategoryID "Unused". Their
names -->
<!-- can be changed. The real taxonomy starts with the root
category. The -->
<!-- unused category is used only by VIC as a sandbox for building
-->
<!-- taxonomies. The Unused category will not appear in a knowledge
tree.-->
<!ELEMENT Taxonomy (RootCategory, Unused?)>
<!ATTLIST Taxonomy
    Name CDATA #REQUIRED
    Version CDATA #IMPLIED
    Created CDATA #IMPLIED
    Modified CDATA #IMPLIED
    Description CDATA #IMPLIED
>
<!-- The Root category may consist of a properties element and a
list of -->
<!-- child categories. Its CategoryID is fixed to "Root".
CategoryID must -->
<!-- be unique. However, the XML ID type is not used to enforce
this -->
<!-- because Verity category ID can contain characters not allowed
in the -->
<!-- XML ID type. -->
<!ELEMENT RootCategory (CategoryProperties?, (Category |
RefCategory)*)>
<!ATTLIST RootCategory
    Name CDATA #REQUIRED
    CategoryID (Root) #FIXED "Root"
    Created CDATA #IMPLIED
```

```
    Modified CDATA #IMPLIED
    Description CDATA #IMPLIED
    SortChildren (Yes|YES|yes | No|NO|no) "Yes"
>
<!-- In VIC 4.5, a category does not contain any concept word or
keyword. -->
<!-- The ConceptWord is added for future use. -->
<!ELEMENT CategoryProperties (ClassifyRule?,
                              AutoCatExtract?,
                              AutoGenCollection*,
                              UserDefined*,
                              ConceptWord*)
>
<!-- A category may consist of a properties element and a list of
child -->
<!-- categories. Its CategoryID must be unique. A category can have
more -->
<!-- attributes than the top-level categories(Root and Unused),
such as -->
<!-- attributes about parents and refining. -->
<!ELEMENT Category (CategoryProperties?, (Category |
RefCategory)*)>
<!ATTLIST Category
    Name CDATA #REQUIRED
    CategoryID CDATA #REQUIRED
    Created CDATA #IMPLIED
    Modified CDATA #IMPLIED
    Description CDATA #IMPLIED
    AutoParent CDATA #IMPLIED
    Refining (Yes|YES|yes | No|NO|no) "No"
    SortChildren (Yes|YES|yes | No|NO|no) "Yes"
    Order CDATA #IMPLIED
>
<!-- A reference category has a CategoryID that refers to the real
category.-->
<!-- A reference category can not have its own children. -->
<!ELEMENT RefCategory EMPTY>
<!ATTLIST RefCategory
    Name CDATA #IMPLIED
    CategoryID CDATA #REQUIRED
    Created CDATA #IMPLIED
    Modified CDATA #IMPLIED
    Description CDATA #IMPLIED
>
```

```
<!-- Same as the Root except that the unused category is used only
by VIC -->
<!-- as a sandbox for building taxonomies. The Unused category will
not -->
<!-- appear in a knowledge tree. Its CategoryID is fixed to
"Unused".-->
<!ELEMENT Unused (CategoryProperties?, (Category | RefCategory)*)>
<!ATTLIST Unused
    Name CDATA #REQUIRED
    CategoryID (Unused) #FIXED "Unused"
    Created CDATA #IMPLIED
    Modified CDATA #IMPLIED
    Description CDATA #IMPLIED
    SortChildren (Yes|YES|yes | No|NO|no) "Yes"
>
<!ELEMENT AutoCatExtract EMPTY>
<!ATTLIST AutoCatExtract
    Source (Path | URL | Meta) #REQUIRED
    Field CDATA ""
    FieldDelim CDATA ""
    HierarchyDelim CDATA ""
    PrefixIgnore CDATA ""
    SuffixIgnore CDATA #IMPLIED
    AutoGenDate CDATA #IMPLIED
>
<!-- VIC 4.5 and ktmgr use a fixed parser for parsing rules in all
the -->
<!-- categories. This may be extended in the future releases.-->
<!ELEMENT ClassifyRule EMPTY>
<!ATTLIST ClassifyRule
    Rule CDATA #REQUIRED
    Parser (Simple | FreeText | BoolPlus | Internet) #FIXED
"Simple"
  Threshold CDATA "0"
  UpperThreshold CDATA "2"
>
<!ELEMENT UserDefined EMPTY>
<!ATTLIST UserDefined
  Name CDATA #REQUIRED
  Value CDATA #REQUIRED
>
<!-- The content model for AutoGenCollection and ConceptWord will
be -->
<!-- defined later. -->
```

```
<!ELEMENT AutoGenCollection (#PCDATA)>
<!ELEMENT ConceptWord EMPTY>
<!ATTLIST ConceptWord
    Word CDATA #REQUIRED
    Weight CDATA #IMPLIED
>
```

Exporting Auto-categorized Taxonomies to XML

Taxonomies that have been auto-categorized save the auto-categorized parents, but not their child categories. The following XML shows a taxonomy file with auto-generated categories by path, metadata, and URL.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Taxonomy SYSTEM "C:\Program Files\Verity\Intelligent
Classifier\common\taxonomy.dtd">

<Taxonomy Name="Root"
          Created="05-May-2003 11:00:00 am"
          Modified="06-May-2003 12:43:05 pm">
  <RootCategory CategoryID="Root"
                Name="Root"
                Created="06-May-2003 12:24:49 pm"
                Modified="06-May-2003 12:24:49 pm">
    <Category CategoryID="C000003"
              Name="example_01"
              Created="06-May-2003 12:24:49 pm"
              Modified="06-May-2003 12:43:05 pm">
      <CategoryProperties>
        <ClassifyRule Rule=" "
                      Threshold="0.0">
        </ClassifyRule>
        <AutoCatExtract Field="URL"
                        Source="Meta"
                        HierarchyDelim="/"
                        PrefixIgnore="http://"
                        AutoGenDate="3135098492">
        </AutoCatExtract>
        <AutoGenCollection>C:\Program Files\Verity\Intelligent
                 Classifier\samples\vrtyweb</AutoGenCollection>
      </CategoryProperties>
    </Category>
```

```
    <Category CategoryID="C000004"
              Name="example_02"
              Created="06-May-2003 12:24:49 pm"
              Modified="06-May-2003 12:43:05 pm">
      <CategoryProperties>
        <ClassifyRule Rule=" "></ClassifyRule>
        <AutoCatExtract Field="DOC_FN"
                        Source="Path"
                        HierarchyDelim="/"
                        AutoGenDate="3135098502">
        </AutoCatExtract>
        <AutoGenCollection>C:\Program Files\Verity\Intelligent
                           Classifier\samples\vrtyweb
        </AutoGenCollection>
      </CategoryProperties>
    </Category>

    <Category CategoryID="C000005"
              Name="example_03"
              Created="06-May-2003 12:24:49 pm"
              Modified="06-May-2003 12:43:05 pm">
      <CategoryProperties>
        <ClassifyRule Rule=" ">
        </ClassifyRule>
        <AutoCatExtract Field="Author"
                        Source="Meta"
                        AutoGenDate="3135098522">
        </AutoCatExtract>
        <AutoGenCollection>C:\Program Files\Verity\Intelligent
                           Classifier\samples\vrtyweb
        </AutoGenCollection>
      </CategoryProperties>
    </Category>

    <Category CategoryID="C000006"
              Name="example_04"
              Created="06-May-2003 12:24:49 pm"
             Modified="06-May-2003 12:43:05 pm">
    </Category>
  </RootCategory>
  <Unused CategoryID="Unused"
          Name="Unused Categories"
          Created="06-May-2003 12:24:49 pm"
          Modified="06-May-2003 12:24:49 pm">
```

```
  </Unused>
</Taxonomy>
```

C

Using Configuration Files

When Intelligent Classifier creates a Knowledge Tree or workspace it also automatically creates some other files that you can use in other Verity applications.

This appendix contains the following information:

Knowledge Base Map (KBM) Files

Intelligent Classifier automatically creates a knowledge base map file (KBM file). It uses the same name as the workspace (`.tsw`) file but with the extension `.kbm`. It stores this `.kbm` file in the same directory as the workspace file.

The KBM file records the paths of the topic sets in the workspace. For example, if a workspace has two topic sets, `Feb01` and `Jan31`, the KBM file might be:

```
$control: 1
kbases:
{
  kb: "Feb02"
    /kb-path = "C:\\Topic Sets\\Feb01"

  kb: "Jan31"
    /kb-path = "D:\\Topic Sets\\Annual Reports\\Jan31"
}
```

Intelligent Classifier only records the names of topic sets that are open in the workspace at the time the workspace is closed. Before you use the KBM file in another application:

- Ensure that all the required topic sets are open in the Knowledge Tree or workspace.
- Close the Knowledge Tree or workspace (or exit Intelligent Classifier).

  This rewrites the KBM file and ensures that it lists all the required topic sets.

If you move the KBM file or topic sets to another machine, you must edit the paths in the KBM file to point to the new locations.

For example, if you create the KBM file on one machine, then move it to use with a K2 server on another machine, you need to ensure that the paths in the KBM are accurate. Either:

- Copy the topic sets with the KBM file to the new machine, and edit the paths to point to their new locations.
- If the topic sets on the previous machine are visible over the file system from the new machine, edit paths in the KBM file to reflect the mapping as seen by the new machine.

  The topic sets need to be accessible over the network.

  The paths can use locations relative to the location of the KBM file.

BIF Files

If you use manual ranking or manual categorization, Intelligent Classifier automatically creates a bulk insert format file (BIF file). It uses the same name as the taxonomy file but with the extension `.bif`, and stores this `.bif` file in the same directory as the taxonomy file.

The BIF file generated by Intelligent Classifier contains entries of the following form:

```
VdkVgwKey:       KEY
AssignCategory: CATA SCORE [CATB SCORE...]
<<EOD>>
VdkVgwKey:       KEY
RemoveCategory: CATA
<<EOD>>
```

KEY is the document's `VdkVgwKey` and *CATn* are category IDs. *SCORE* is a number between 0 and 65536 and is used to rank the document within the category. A value of 10000 corresponds to a score of 1.0. Manually ranked and manually categorized documents (which are always at the top of the results list) have score values greater than 10000.

For example:

```
VdkVgwKey:       docs/demo_templates/searchresults.html
RemoveCategory:  reports
<<EOD>>

VdkVgwKey:       docs/products/keyview/index.html
AssignCategory:  annual 10001 reports 10001
<<EOD>>

VdkVgwKey:       docs/products/ko/faql.html
AssignCategory:  reports 10002
<<EOD>>
```

For more information on BIF files, see the *Verity Collection Reference.*

You can also import a BIF file that specifies manual assignments.

Collection Map (CLM) Files

Intelligent Classifier automatically creates a collection map file (CLM file). It uses the same name of the taxonomy (`.tax`) file but with the extension `.clm`, and stores this `.clm` file in the same directory as the taxonomy file.

Adding and Removing Collections

Knowledge Trees and workspaces contain references to one or more *collections* that specify which documents will be searched.

1. To add a collection, choose either **File | Add/Remove Collections**, or click the toolbar icon. The Collections dialog box appears.
2. Select the collection. Either select one of the recently used collections and click **Add** to add the collection or select **Remove** to remove the collection from this knowledge tree. This step does not delete a collection.

   The remainder of the information in this section applies only to adding a collection.

   If the collection that you want to add is not in the list, click Browse and select the collection using the file browser.

   Note Intelligent Classifier only enables the browser's OK button when a collection's directory is selected. When ordinary directories are selected it disables the button.

3. Click **OK** to close the Browse dialog.

Add Options

Optionally, you can select the location of the documents referenced in the collection.

1. Click **Properties**. The Collection Properties dialog appears.
2. Choose how the documents will be retrieved when they need to be displayed in the Document View Pane:
   - If the documents are stored locally or on a networked drive, select **File system**.

- If you do not have file system access to the documents and do *not* have access to a K2 Server installation that holds that collection, or a copy of it, then select **Recreate from collection**. In this case, Intelligent Classifier does not access a document directly but *recreates* it from the word list in the collection. The document is displayed as unformatted text because the collection does not store any formatting information.
- If you do not have file system access to the documents but *do* have access to a K2 Server installation that holds that collection, or a copy of it, then select **Server**. In this case Intelligent Classifier retrieves the documents from the K2 Server and can display them properly formatted.

  Type the name of the computer hosting the K2 server and the port number. Usually the Collection Name can be left blank. Fill it in with the name of the collection as specified in the K2 server if you need to resolve ambiguity about which collection to access.

3. Click **OK** to close the Collection Properties dialog.
4. If the collection is a *secure collection* (that is, an ODBC collection, Lotus Notes collection, or Microsoft Exchange collection) an authentication dialog appears.
5. Type the required information to access the secure collection.
6. Click **OK** to close the authentication dialog.
7. Click **OK** to close the Collections dialog box.

If you add a secure collection to a Knowledge Tree or workspace and then close that Knowledge Tree or workspace, you will need to use the **File | Authenticate** command to log back in to the secure collection when you reopen the Knowledge Tree or workspace.

Using CLM Files

CLM files cannot be used on NT when adding collections to a workspace. You must use the following steps:

1. Select **File | Add/Remove Collections**.
2. Select **Browse** and navigate to each collection and select it to bring it into Intelligent Classifier's current workspace.

D

Valid Verity Query Language Operators and Modifiers

The following operators and modifiers are valid for Intelligent Classifier.

Table D-1 Valid VQL Operators and Modifiers

Operators	Advanced Operators	Modifiers
ACCRUE	COMPLEMENT	CASE
ALL	LOGSUM and LOGSUM/n	LANG/ID
AND	MULT/n	MANY
ANY	PRODUCT	NOT
BUTNOT	SUM	ORDER
IN	YESNO	WHEN
NEAR and NEAR/n		
OR		
PARAGRAPH		
PHRASE		
SENTENCE		
SOUNDEX		
STEM		
THESAURUS		

Table D-1 Valid VQL Operators and Modifiers (continued)

Operators	Advanced Operators	Modifiers
TYPO/n		
WILDCARD		
WORD		

For detailed information about the operators and modifiers, see the *Verity Query Language and Topic Guide*.

Intelligent Classifier also supports a <FIELD> operator, which is not a true operator but can be used to preface a field name. Using <FIELD>, you can use the following relational operators:

- CONTAINS
- ENDS
- MATCHES
- STARTS
- SUBSTRING

Intelligent Classifier supports the <INFLECT> operator. The INFLECT operator is a linguistic expansion operator that supports some locales, such as the AMA (Arabic) locales. The INFLECT operator is used in place of the STEM operator.

E

Getting Started with the Quick Start Guide

This appendix is a quick–start guide for keystrokes within Verity Intelligent Classifier. Table E-1 only covers those keystrokes that include the Ctrl, Alt, or function keys.

Table E-1 Keystrokes for Intelligent Classifier

Keystroke	Action
Ctrl + N	New Workspace
Ctrl + S	Save All
Ctrl + O	Open Knowledge Tree
Ctrl + V	Paste
Ctrl + X	Cut
Ctrl + C	Copy
Ctrl + F	Find
Alt + Enter	Node Properties
Ctrl + R	Run Query
Ctrl + T	Test Topic
Ctrl + Q	Query Input Mode
Ctrl + T	Test Category
F1	Intelligent Classifier Help
F5	Add Top Level Node
F6	Add Sibling Node
F7	Add Child Node

Glossary

term	definition
API (Application Programming Interface)	A set of routines that an application uses for creating programs that interface to other programs.
authentication	The process of passing credentials to a secure server, such as an Information Server or a K2 Server, K2 Broker, or K2 Ticket Server.
BIF (bulk insert file)	A text file used to compile data into Verity internal formats. For example, a BIF is used to submit documents to a collection for indexing, or to insert queries into a Profile Net. Typically, BIFs are also used to insert or edit metadata not found in the source documents being indexed.
branch node	In Verity Intelligent Classifier, a node that is between the top-level nodes and the leaf nodes.
browse	A command-line tool that lists the contents (field names and values) of a collection's document table.
browsing	In K2, the act of traversing a knowledge tree to view the contents of a collection category-by-category. See also category drill-down.
bucket	A container for all keys associated with a discrete value or range in a parameter in a parametric index; it identifies a set of like documents for parametric selection.

bucket set	The set of all buckets associated with a parameter in a parametric index.
case-insensitive search	A type of search in which the case of the letters in the search term does not matter. In case-insensitive search, the search term *Cat* would find all instances of *cat* or *CAT* or *Cat*, for example. Conversely, in case-sensitive search, the search term *Cat* would find only instances of *Cat*.
category	A subject of interest used to collate documents that are relevant to the subject; logically, a category represents a node in a taxonomy.
categorization	See classification.
category alias	An alternative name used to identify the category in all parts of the system.
category definition	A mathematical rule against which a document can be evaluated for membership in the category.
category drill-down	The process of successively browsing from a category to one of its sub-categories to find information.
category ID	A unique identifier for a category.
category name	The name or label associated with a category, used when browsing a knowledge tree or results list. Category names do not have to be unique.
character map	A mapping between characters encoded in one character set and the same characters encoded in a second character set. Often, different character sets encode the same characters, because different standards were set up by different groups of people to encode the same set of characters from one character set to another without loss.
character set	A numeric encoding of the characters of a language. Text in a given language can be stored and manipulated using one or more character sets. The mapping occurs between characters and byte strings; that is, the combination of a particular character encoding (which maps between byte strings and integers) and a particular coded character set (which maps between integers and characters). Examples include ASCII, Shift-JIS, and UTF-8.
charset	In general terms, this is an abbreviation of character set. This term corresponds to the `-charset` argument, which sets a character set using a given command-line tool.
child node	In Verity Intelligent Classifier, a node directly underneath another node in the topic tree.

classification	The process of assigning documents to categories in a taxonomy. See also content organization.
cluster	A group of documents related by similarities in their content.
clustering	The process of automatically discovering the clusters in a set of documents.
collection	The set of files and folders that stores all the information needed by VDK to search and classify documents in a repository. A collection stores the locations of all the indexed documents, the locations of all the indexed words in those documents, and metadata about the documents. It does not store the documents themselves.
collection schema	The fields internal to a collection (as specified in `style.ddd` and any optional `style.ufl` files) and fields external to the collection (as specified by the gateway).
collection-level security	A security system that controls access to collections based on the user's identity. See also: document-level security.
concept extraction	The analysis and extraction of recurring key concepts in documents, and the process of relating these key concepts to the particular documents containing them.
concept mapping	The process of automatically extracting the key concepts contained in a set of documents and organizing them into a hierarchy, which is called a concept tree.
concept tree	A hierarchy of key concepts, which is generated by concept mapping.
content organization	The process of building taxonomies, defining categories, and populating taxonomies. See also classification.
corpus	A set of documents drawn from one or more repositories and indexed by K2.
default installation locale	The locale specified in the configuration file `verity.cfg`. If defined, it is the default session locale.
default session language	The language used as the default for queries during a VDK session. This applies only when the session locale is the multilanguage locale (`uni`).
default session locale	The locale assigned to a VDK session if no locale is specified when the session is opened.

delimiter

A character used by the tokenizer to split document text into searchable units. For many locales, white space and punctuation are the most common delimiters. See also tokenization.

document filter

A driver-level plug-in software module that can read documents in one or more specific formats (such as PDF, XML, or Microsoft Word). Document filters receive documents from gateways, extract text data and field information from them, and pass that information along for indexing and storage in a collection.

document key

A unique identifier for each document in a collection. It identifies the gateway key and collection alias of a document.

document profile

An internal representation of a document. It is based on:

- the content of the document
- the relevance of the document content to query terms

Document profiles are created on a per-collection basis, and are initially seeded with the information stored about the document in a Verity collection.

document table

A table in a collection that specifies the location of each indexed document. The document table also contains all metadata (fields) associated with each document.

document-level security

A method of controlling which documents appear in search results for a particular user.

See also: collection-level security.

double-byte string

A string formed by a sequence of characters from a double-byte character set. Each character in a double-byte character set is strictly two bytes in length. The null character that terminates the string is also a two-byte null-character. A double-byte string differs from a multibyte string in that the characters are all a fixed length of two bytes, whereas the characters in a multibyte string are a variable length. Unicode is an example of a double-byte character set.

dynamic highlighting

A method of highlighting the search term in a document summary or in a retrieved document. In dynamic highlighting, the application actually searches through the results or the document to locate and highlight the term. Dynamic highlighting is slower but more accurate than static highlighting.

extended character

1. A character above the ASCII range (32 through 127) in Windows-based single-byte character sets.
2. An accented character.

external fields	Repository field names exposed to the Verity search engine. External field names can be the same as repository field names or an obvious mapping of the repository field name. See also: internal fields.
field	A discrete data item associated with the document such as the author, title, document location, creation date, and so on.
foreign word	In the `uni` locale, a word in any language other than the overall language of its document.
full-width character	In Japanese, a Katakana or Romaji character that occupies the same amount of horizontal space as a Kanji character. In Japanese character sets, a full-width character has a different character code than its half-width equivalent.
fuzzy search	The ability to find and retrieve a document even with spelling and typographical errors in the search. Phonetic and sounds-like searches are also possible.
gateway	A set of access methods, optimized for use by the Verity search engine, used to retrieve a document from a repository for both indexing and viewing. For example, administrators use the Exchange Gateway to index Microsoft Exchange documents. K2 includes gateways for local file systems, the web, Documentum, ODBC (Oracle, and so on.), NNTP (NetNews), RTI (Newswire), MAPI (MS Exchange), and Notes. New gateways can be created with the Verity Gateway Developer's Kit (GDK).
gateway key	A unique identifier for any document or record in a collection. It is used to locate and retrieve the contents of the document. It is also known as the `VdkVgwKey`, short for Verity Development Kit Verity Gateway Key.
half-width character	In Japanese, a Katakana or Romaji character that occupies half the horizontal space of a Kanji character. In Japanese character sets, a half-width character has a different character code from its full-width equivalent.
i18n	An abbreviation for internationalization.
index	A Verity collection, knowledge tree, or parametric index.
index file	A file used by a search engine to locate specific web pages in a web site. The structure of an index file is similar in concept to the index of a book, where keywords are cross-referenced to their occurrence on pages.

indexing	A process that scans each document to be indexed and enters document text words and locations, as well as the metadata (title, author, size, internal zones, and so on) into a collection.
internal character set	See session character set.
internal fields	Document fields that are stored in indexed document tables internally managed by the Verity search engine. In contrast, external fields are stored in indexed document tables in external repositories, such as applications, relational databases, and so on. Internal fields are stored in the `style.ddd` file associated with a collection. Internal fields can be displayed in a results list or used to define how the document body text is viewed. Only internal fields, not external fields, can be searched using Verity Query Language operators.
internal locale	See session locale.
internationalization	The process that occurs during application development that makes localization easier by separating locale differences from the rest of the program, which stays the same. If internationalization is thorough, localization requires no programming.
Internet-style query parser	A free-text query parser that lets users conduct familiar web-style searches.
iterator	Software used to "walk" an arbitrary string, one character at a time. It recognizes that multibyte characters vary in length and can jump ahead several bytes at a time. It keeps some state information when walking the string so that it can recognize the difference between one-byte and two-byte mode. This allows it to return characters from the string one at a time, no matter how many bytes are in each character.
K2 Broker	A K2 service that receives client search requests and distributes them to available K2 Servers.
K2 Dashboard	A browser-based user interface that enables administrators to view and change configuration settings for K2 services from a single computer, even when the K2 services reside on many different computers.
K2 domain	A K2 system consisting of one Master Administration Server and all the K2 services (K2 Ticket Servers, K2 Brokers, K2 Servers, and so on.) configured by that Master Administration Server. Note that a K2 domain is unrelated to a Windows NT domain. See also: K2 system.

K2 search group	A set of K2 Brokers and K2 Servers containing one top-level K2 Broker and all the other K2 Brokers and K2 Servers attached below it, possibly including ones in different K2 domains. Therefore, a search request handled by the top-level K2 Broker can be passed to any of the other K2 Brokers and K2 Servers in the search group, including those from other K2 domains.
K2 Server	A K2 service that receives search, viewing, profiling, and recommendation requests and performs searches of collections, knowledge trees, and parametric indexes.
K2 services	The executable processes in a K2 system, such as a K2 Broker, a K2 Server, or a K2 Ticket Server.
K2 system	A generic term meaning a K2 installation. It may be either a K2 domain or a K2 search group.
K2 Ticket Server	The K2 service that is used to implement document-level security and (along with the gateway) to implement collection-level security.
K2User	The information obtained via a K2UserLogin call from a K2 Ticket Server. It contains encrypted user ID information and encrypted server specifications.
KeyView filter	A document filter, based on Verity KeyView technology, that is used during indexing to process many types of files.
knowledge tree	A structure for organizing documents for navigation to subjects of interest. A knowledge tree consists of a taxonomy, category definitions, documents that have been placed into categories within the taxonomy by applying category definitions, and other information about the documents.
l10n	An abbreviation for localization.
language ID	A two-character (ISO 639) code that specifies an individual language. Examples are `en` for English and `zh` for simplified Chinese. Verity uses language IDs for specifying languages for the multilanguage locale and the language identification command-line tool.
language identification	A Verity command-line tool that identifies the language of a document.
language identification filter	A document filter (`flt_lang`) used by the multilanguage locale to assign a language to a document before indexing.
LCID	See locale ID.

leaf nodes	In Verity Intelligent Classifier, the nodes that form the lowest level of the topic tree.
locale	A geographic or political region that shares the same language and customs. See also Verity Locale.
locale definition file	A file (`loc00.lng`) in each locale's directory that controls the language handling characteristics of the locale.
locale ID (LCID)	A 32-bit value defined by Windows that consists of a language ID, a sort ID, and reserved bits.
locale-sensitive	Exhibiting different behavior or returning different data, depending on the locale. For example, the Win32 sort functions return different results depending on the locale parameter sent to each function.
localization	The process of adapting a program for a specific international market, which includes translating the user interface, resizing dialog boxes, defining lexing and stemming rules, customizing features (if necessary), and testing results to ensure that the program works as expected. See also: internationalization.
(LRC) Logistic Regression Classifier	Software that creates a category definition from a set of positive and negative exemplary documents. Positive documents refer to documents that are relevant to the topic (or category) of interest. Negative documents are the opposite, that is, documents that are irrelevant to the topic (or category) of interest.
map	Provides a way of limiting the repository fields available to the Verity search engine. The gateway can specify external fields and then map them to the repository fields they represent.
Master Administration Server	The central hub for K2 configuration information. A K2 domain must have one and only one Master Administration Server. (A Master Administration Server is an "Administration Server" but you must use the proper name to distinguish it from the other Administration Servers in a K2 domain. Do not use "local" or "remote" or any other such modifier. Refer directly to a Master Administration Server and the other Administration Servers that exist in a K2 domain.)
metadata	Data that describes other data. For example, `Author` and `Size` could be metadata for a Microsoft Word document. Most metadata can be used in a Verity Query Language expression to search for documents containing the given metadata value.
`mktopics`	A command-line tool for building and updating topic sets.

`mkvdk`	An all-purpose command-line collection maintenance tool.
modifiers	Query terms used in conjunction with operators to change the standard behavior of an operator.
multibyte string	A string formed from characters encoded with a multibyte character set. A multibyte character set uses a variable number of bytes to represent one character.
multilanguage locale	A Verity locale (`uni`) that supports multiple languages simultaneously. See also **single-language locale**.
named nodes	In Verity Intelligent Classifier, a node that has been given the name that distinguishes it from all other nodes in the topic set. Equivalent to a topic.
node	In Verity Intelligent Classifier, an element in the topic tree. A node can be an unnamed node, or a named node. Named nodes represent topics.
noun phrase	A group of words (for example, *due process* or *court of law*) that functions as a noun. Part-of-speech processing during indexing can lead to the automatic extraction of noun phrases, which can be used in the automatic creation of document features and summaries.
ODBC (Open Database Connectivity)	A middleware software standard that allows an application to communicate with various database engines.
ODK (Organization Developer's Kit)	An SDK (Software Development Kit) containing APIs that enables developers' applications to create and populate taxonomies. This SDK provides similar functionality to Verity Intelligent Classifier.
operators	Reserved words that describe the relationship between search terms in the Verity Query Language.
OTL file	A text file containing a representation of a topic set.
parameter	In parametric selection, a set of discrete values or ranges.
parametric index	An index that enables retrieval of documents based on the values of parameters.
parametric selection	The ability to search for documents based on a value or values of one or more parameters. Parametric selection can be combined with full-text search on document content.
parent node	In Verity Intelligent Classifier, a node directly above another node in the topic tree.

partition	A subdivision of a collection. Partitioning collections improves scalability and searching performance.
part-of-speech processing	During indexing, the assignment of the appropriate part of speech (noun, verb, adjective, and so on) to each token in the word index.
primary document key	See document key.
Profile Net	The set of stored queries against which a Profile Service evaluates documents.
Profile Service	A service that processes large numbers of queries into Profile Nets and evaluates the incoming stream of documents against these queries. Developers can use Profile Services in applications such as message routing, document tagging, and classification.
purging	The act of deleting all records from a collection.
`rcadmin`	A command-line tool used to administer K2. It has similar functionality to the K2 Dashboard.
`rck2`	A command-line tool used to connect to K2 Servers for searching collections and other Verity indexes.
relational taxonomies	A set of taxonomies containing the same documents, supporting simultaneous navigation through all the taxonomies. See also taxonomy.
repository	A group of documents that are all stored in the same location, such as a file system. Other repositories can be organized in a relational database or a proprietary storage system such as Microsoft Exchange folders or Lotus Notes databases.
scoped search	A search limited to documents in specified **categories**.
score	A numerical value indicating the degree of match between a document and a query. Scores, usually expressed to the end user as a decimal number between 0 and 1, are calculated during Verity search or Profiler operations. Scores are based on numerous factors, including the number of times search/query words appear in the document, their location in the document, and their proximity.
SDK (Software Development Kit)	A set of APIs and related tools that enables a developer to create applications.
search application	The K2 components enabling administrators to index content and end-users to search and view that content. A search application typically includes K2 services and Verity collections.

search engine	A software application that queries an index file. In broader terms it refers to a collection of programs which are used to index a repository and then present a user interface that enables queries to be constructed and searched.
session character set	The character set used for input to and output from VDK during a VDK session. It must be a character set supported by the session locale.
session locale	The locale used for all operations during a VDK session.
sibling node	In Verity Intelligent Classifier, a node at the same level as another node, and directly below a parent node.
simple tokens	A behavior, available for some locales, in which nearly all symbols (in addition to white space and punctuation) are defined as delimiters. In simple-token behavior, words are broken down into smaller searchable units, thus increasing the potential for search hits.
single-byte string	A string written in a single-byte character set. Each character in a single-byte character set is one byte long. Single-byte strings can be 8-bit strings that use extended characters, or 7-bit strings that use ASCII.
single-language locale	A Verity Locale that supports only one language. Most locales are single-language. Compare multilanguage locale.
sorting order	The order in which a locale sorts the characters of its language. Verity Locales sort characters in a manner that facilitates accent-insensitive and case-insensitive search and display.
Soundex search	A type of search in which occurrences of the search term plus any words with similar pronunciation are returned. Verity supports Soundex search for the English language only.
static highlighting	A method of highlighting the search term in a document summary or retrieved document. In static highlighting, the application uses offsets in the collection's word index to calculate the positions of terms to highlight. Static highlighting is faster but less accurate than dynamic highlighting.
stemmed search	A type of search that locates all words that share the same word stem. For example, a stemmed search for the term *house* would find all occurrences of *house*, but also all occurrences of *houses*, *housed*, and *housing*.
stop word	A search term that should be ignored. Verity supports several types of stop-word lists, some used at indexing time and others used at search time.

style file	A file used to configure the series of indexes in a collection that store data about its documents. The choice of a particular style file determines how indexing utilities function.
`style.dft`	A collection style file that controls the contents of the virtual document created during indexing.
`style.fxs`	A collection style file that contains feature-extraction stop words, that is, words that should not appear in document summaries and clusters. See also `vdk30.stp`.
`style.lex`	A collection style file that can control how tokenization occurs during indexing. Use of `style.lex` is discouraged; tokenization control is now available through the locale definition file associated with each locale.
`style.prm`	A collection style file containing parameters that control the generation of specialized indexes.
`style.stp`	A collection style file that contains indexing stop words, that is, words that should not be included in the collection's word index.
`style.ufl`	A collection style file that defines custom fields to be included in the collection's document table and optionally specifies the generation of indexes for those fields.
`style.uni`	A collection style file that controls the functioning of the universal filter.
`style.zon`	A filter style file that controls functioning of the zone filter.
styleset	The complete set of style files.
StyleSet Editor	The Verity application that enables administrators to create and modify style files.
synonym search	A type of search that returns all occurrences of the search term and also any of its synonyms, as defined in a thesaurus.
system default locale	The default session locale if the default installation locale is not defined.
TAX file	A taxonomy file. This stores the taxonomy in a text format.
taxonomy	The hierarchical organization of data by category. A taxonomy defines the view(s) by which administrators want to organize data.
thematic mapping	Maps the key concepts in a collection of documents and the relationships (parent-child, sibling, and so on) between them.

thesaurus	A dictionary of synonyms. Each Verity Locale supports use of a thesaurus for searching. In a synonym search, all occurrences of the search term and any of its synonyms are returned.
token	A searchable unit in a document. Tokens are typically the individual words in a document, but they can also be word stems, or any string fragments that occur between delimiter characters.
tokenization	The process by which the tokenizer converts a document's text into searchable units (such as words and word stems). The tokens are then stored in a collection's word index.
top level topic	The topic at the top level of the topic tree.
topic	A stored query expression written in the Verity Query Language (VQL) that is used (1) to model a concept of interest in a classification task, or (2) to enable users to quickly find information without having to compose sophisticated queries using complex syntax in a search task. In a classification task, a topic can be used by itself, or combined with other topics to specify a category definition.
topic set	A grouping of topics that have been compiled for use by a Verity application; for classification tasks, a topic set contains one or more topics used to classify documents in a collection.
topic tree	The hierarchy of the topic set. It is shown in the Topic Pane of Verity Intelligent Classifier.
typo search	A type of search that corrects for minor misspellings in the search terms. In a typo search, occurrences of the search term and any words close to it in spelling are returned.
Unicode	A standard double-byte character set. The Unicode standard encodes the characters for all major modern languages. The advantages include characters that are always a fixed sized (2 bytes), and all characters can be represented in one character set. There are various implementations of portions of the Unicode standard. The implementation used by the Verity multilanguage locale is UTF-8.
universal filter	A document filter that receives raw data from the gateway and determines the file type of the incoming document. Based on this file type, the filter invokes a suitable helper filter, which extracts the available text and metadata from the document.

user profile	An internal representation of a user. A user profile is created over time from information such as the following: ■ documents authored by the user ■ interests submitted ■ queries asked ■ documents rated or viewed
VDK	Can either mean: 1. Verity Developer's Kit, the API that enables OEM developers to build Verity functionality into their products 2. the programming core on which most Verity applications are built.
`vdk30.stp`	A locale-specific file that contains feature-extraction stop words, that is, words that should not appear in document summaries and clusters. See also `style.fxs`.
`VdkVgwKey`	See gateway key.
Verity Developer's Kit	See VDK.
Verity Intelligent Classifier	An application for creating, viewing, editing, and testing topics and taxonomies.
Verity Locale	A software module that allows Verity applications to operate on documents in a specific language or set of languages. A locale provides one or more capabilities that may include tokenization, stemming, part-of-speech recognition, and thesaurus use. See also single-language locale and multilanguage locale.
Verity Query Language (VQL)	Verity's standard language for issuing searches consisting of operators.
weight	In the Verity Query Language, a number that can be used to represent the importance of different parts of the query. Weights are combined with the scores returned by the query's operators and hence affect the documents' scores.
wildcard search	A type of search in which the search term contains special symbols that represent multiple characters. For example, a wildcard search with the term *abc** returns occurrences of all words that start with *abc*.
word index	Stored in a collection, a list of all words that appear in the documents, plus the location of every instance of the word.

XML (eXtensible Markup Language)	A simplified form of SGML. A W3C standard for semantic and structural tagging of XML documents. It is a set of rules for forming semantic tags that break a document into parts and identify the different parts of the document.
XML filter	A document filter that processes XML documents.
XML schema	A structured framework or plan that contains elements or tags and their definitions to outline the organization of XML file content.
zone	A named region of a document that can be searched. Examples are HTML tags such as `H1`, `H2`, `BODY`, `TITLE`, and so on., or the values of the `TO`, `FROM`, and `SUBJECT` fields in email and Usenet messages. The Verity Query Language syntax is `text <IN> zoneName` Note that some information can be (and often is) saved as both a field and a zone (the document title, for example).

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X