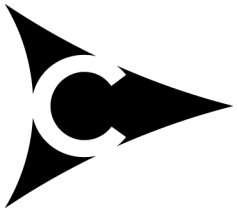

CLARIT Routing Programming Guide

Clairvoyance Corporation

January 30, 2001

Warranties and Liabilities

The information in this document is subject to change without notice and does not represent a commitment of Clairvoyance Corporation. Clairvoyance Corporation assumes no responsibility for any errors that appear in this document.

Trademarks

CLARIT is a registered trademark of Clairvoyance Corporation.
UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.
Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Intellectual Property Protection Notice

The documentation contained herein is the intellectual property of Clairvoyance Corporation. It is protected under the patent, copyright, trademark, and trade secret laws of the United States and the Commonwealth of Pennsylvania.
The contents include proprietary information which, if disclosed, would cause serious adverse consequences for Clairvoyance Corporation. The contents may not be reproduced in any fashion or any medium, except as may be expressly permitted by licensing or other contractual provisions granted expressly and in writing by Clairvoyance Corporation. No disclosure, reverse engineering, or other indirect reproduction may be made without the express, written permission of Clairvoyance Corporation.

Clairvoyance Corporation
5301 Fifth Avenue
Pittsburgh, PA 15232

Phone: +1.412.621.0570
Fax: +1.412.621.0569
Internet: info@clairvoyancecorp.com

Contents

Chapter 1

CLARIT Routing Overview

The CLARIT Routing system classifies information. Documents from an existing collection or from a real-time document stream, such as a news feed, are matched to interest profiles. An interest profile corresponds to a category of documents and defines criteria a document must meet to match that profile, or in other words, to be assigned to that category. Some example applications include the following:

- Sending news stories of interest from a news feed to a user
- Routing e-mail to folders or interested recipients
- Categorizing technical reports against Chemical Abstract Society keywords
- Tagging newspaper articles with predefined subjects

Routing appplications typically select documents from a stream of changing documents based on user-specified interests and send these documents to a destination like an email address or folder. Tagging applications annotate documents in a static database with the category they belong to, based on a controlled vocabulary.

In a CLARIT system, queries, documents, and interest profiles are all documents. In CLARIT terminology, a document represents a collection of text defined by the user. CLARIT indexing builds a database from any collection of items that can be viewed as documents, such as interest profiles. Thus, the reversal between retrieval and routing is purely one of perspective.

CLARIT Routing builds on the standard CLARIT retrieval functionality. When searching for information (the normal information retrieval process), a set of documents comprises the database against which incoming queries are matched.

In a routing application, the incoming documents play the role of the queries, while the interest profiles play the role of the static database. In effect, CLARIT scoring involves matching documents against documents; sometimes, query documents are matched against static database documents, other times, documents to be routed are matched against static interest profiles.

However, there is one significant difference between CLARIT retrieval and routing. For retrieval, the statistics used for scoring are computed from the documents in the database. For routing, the statistics used for computing relevance scores between documents and indexed interest profiles must come from documents other than the interest profiles themselves; otherwise, the contents of one interest profile would effect the retrieval performance of other interest profiles

Routing applications require a set of "computation" or "reference" statistics for computing relevance scores. These statistics are from a reference database and managed by the profile set. To achieve good document scoring, the documents providing the statistics must be similar to the documents being routed, so that terms have similar frequencies and distributions. This means that if you are routing multiple data sources, each one should have its own reference statistics, which correspond closely to that source's unique usage and terminology.

CLARIT Routing uses statistics in two areas:

- Ranking documents as to how well they match a profile. This depends on scoring the documents, which requires statistics that reflect the documents being routed.
- Setting the relevance threshold. This is essentially a threshold for a profile that a document score must exceed in order to be considered relevant to that profile. A document is assigned to a profile based on the threshold. Setting a profile's threshold is critical for performance.

When tagging or categorizing a static database of documents, the statistics for that database are usually used as the reference database. When routing a changing document stream, the statistics from the reference database must match the incomining documents as closely as possible.

This overview contains the following sections:

- Routing Architecture
- Routing Applications
- Writing a Routing Application
- Notes on Building and Running a Routing Application

Routing Architecture

CLARIT Routing provides the **InterestProfile**, **ProfileSet**, **MatchedProfiles**, and routing policy classes to construct routing and tagging applications. The actual routing process is straightforward. As Figure 1-1 on page 3 illustrates, incoming documents are routed against a **ProfileSet** of **InterestProfile**s, resulting in **MatchedProfiles**. The **MatchedProfiles** can be iterated over, retrieving each matching profile so the document can be delivered to the destination designated by the profile.

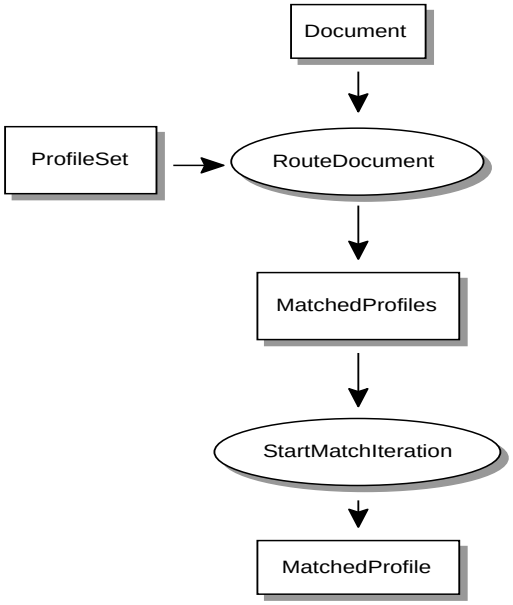

Figure 1-1 Document Routing Methods

This section discusses the following classes:

- **InterestProfile**
- **ProfileSet**
- **MatchedProfiles**
- **Routing policy classes**

InterestProfile

An interest profile is a document containing terms characterizing an interest or category. Each interest profile has an associated threshold indicating the required strength of similarity in order for an incoming document to be assigned to that interest profile or tag.

The **InterestProfile** class represents an interest profile. An interest profile resembles a query in that it can be used to compute a relevance score against other documents. Profiles have a threshold. If a document's relevance score exceeds the profile's threshold, the document is said to "match" the profile. By thresholding the relevance score, an application can decide whether the document should be associated with the interest profile. That is, if the profile represents a destination, then the document can be routed to that destination; similarly, if the profile represents a classification, then the document can be tagged with that classification. A profile's threshold is calculated using the reference statistics for the profile set.

Each interest profile has six fields:

- a title, which names the interest profile
- a text abstract, which is the text that forms the profile's "query". This is parsed to identify the terms, which are stored in the body field.
- a body that contains the weighted terms
- a string specifying the ‘‘destination’’,
- the relevance score threshold for routing or tagging an incoming document,
- a Boolean constraint on the incoming document's field values that must be satisfied for it to be routed or tagged

The **InterestProfile** class provides methods to retrieve and add all of these fields, methods to translate between interest profiles and queries, and a method to commit changes made to a profile

ProfileSet

Multiple interest profiles can be indexed together in a **ProfileSet**. This allows an application to route or tag a document against many profiles efficiently.

The **ProfileSet** class represents a **Corpus** of interest profiles. Compared to normal retrieval, an interest profile corresponds to a query. CLARIT indexing,

however, indexes any collection of items that can be viewed as documents. Queries, of course, are documents; thus, CLARIT can index interest profiles.

By storing multiple interest profiles in a single database, an application can route or tag (i.e., classify) a given document against all interest profiles simultaneously. The incoming document during routing or tagging, then, plays the role of a query in normal retrieval.

The statistics used for computing relevance scores between incoming documents and indexed interest profiles, however, must come from documents other than the interest profiles themselves; otherwise, the contents of one interest profile will have an effect on the retrieval performance of other interest profiles. Thus, a **ProfileSet** will maintain a set of ''reference'' statistics for computing relevance scores. These reference statistics usually come from a database of documents similar to the documents being routed. For tagging applications, the statistics are usually from the actual database of documents being tagged.

MatchedProfiles

A **MatchedProfile** represents the association of an **InterestProfile** with a score that reflects its match against an incoming document that is being routed or tagged.

The **MatchedProfiles** class is a set of **MatchedProfile** instances. The relevance score of each matched interest profile exceeds its score threshold. Each instance keeps track of the profile set that was used to produce these results. This class provides iterators to enumerate the documents matching the interest profiles or the profiles matching a document.

Routing Policies

The CLARIT Routing module has several capability classes and subclasses illustrated in Figure “Routing Policy Classes” on 6. These classes provide methods to route documents and to set statistics on a profile set. As with most CLARIT modules, capabilities are provided for processing either CLARIT **Document** instances or streams of formatted text.

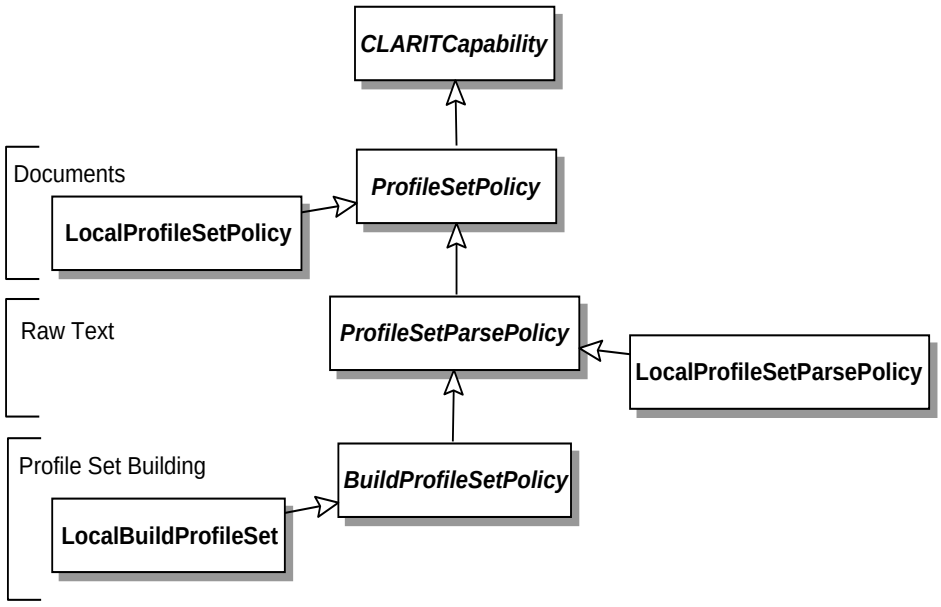

Figure 1-2 Routing Policy Classes

Figure 1-3 on page 7 illustrates routing using the methods provided by routing policies. Policy classes return a **MatchDispatch_Iterator** when routing documents. This iterator returns a **Document** and an iterator for all profiles matching that document.

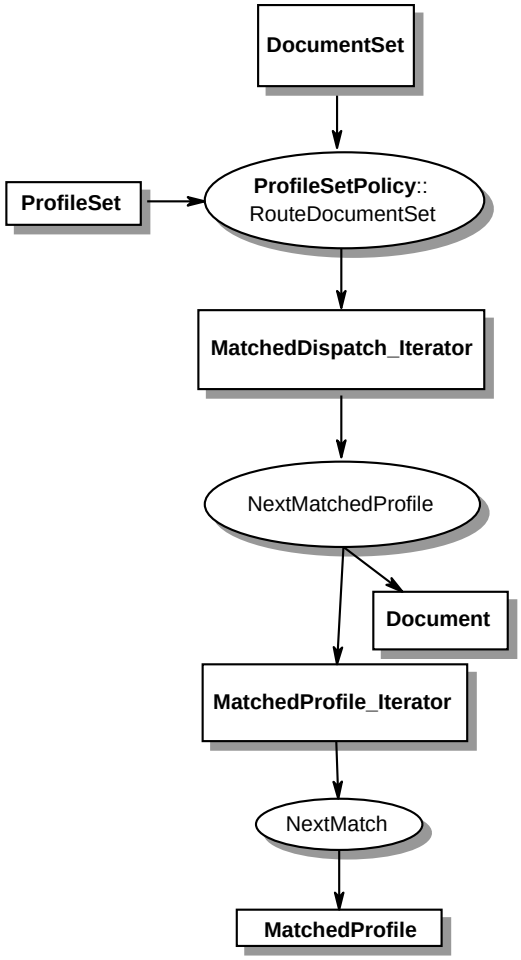

Figure 1-3 Methods Provided by the Routing Policies

Routing CLARIT Documents: ProfileSetPolicy

If instances of a CLARIT **Document** are available, use the **ProfileSetPolicy** capability subclass to route the documents. A **ProfileSetPolicy** instance, created using its concrete subclass (**LocalProfileSetPolicy**), provides methods for routing documents, setting computation statistics, and committing interest profiles to the profile set.

The routing methods return a **MatchDispatch_Iterator**. This nested iterator enumerates each routed document and a iterator that enumerates each interest profile that was matched by that document. Programs use the **NextMatchedProfile** method to iterate through the routing results.

Routing Formatted Text: ProfileSetParsePolicy

If you have a stream of formatted text, use the **ProfileSetParsePolicy** capability subclass to route documents. A **ProfileSetParsePolicy**, created using its concrete subclass (**LocalProfileSetParsePolicy**), provides methods to create a standard field plan. This field plan can be modified to control the fields (parts of a document) that should be analyzed when routing (see **CreateStdFieldPlan**). This class can also create a scanner appropriate for understanding the particular text format for the input stream of documents (see **CreateFieldScan**), if the text does not use standard CLARIT markup.

Like **ProfileSetPolicy**, **ProfileSetParsePolicy** provides methods to route documents and reset statistics for the profile set.

Building a ProfileSet: BuildProfileSetPolicy

If your application will build its own profile set, instead of creating an empty profile set using **prbuild**, you need a **BuildProfileSetPolicy** instance. A **BuildProfileSetPolicy** instance, created using **LocalBuildProfileSet**, provides methods to build an empty profile set. Once built, you must associate statistics with the profile set before routing.

Environment Options

Environment options are declared in **ProfileSetEnvironment**. Applications can use these options to control CLARIT routing. These options include flags that control parsing, scoring, and display. See the **ProfileSetEnvironment** reference page for a full description.

Routing Applications

The CLARIT Routing API is used to develop applications that involve one or more common document classification tasks: document routing, document filtering, document categorization, and document tagging. All of these tasks involve classifying documents into document categories and, in practice, the distinction among them is often so subtle that they are interchangeably and inconsistently named. The CLARIT Routing API is used to develop all types of classification applications, even though its name refers to only one type of application.

Types of Routing Applications

Type	Classification Objectives	Document Categories	Document Source
Document Routing	Grouping and viewing documents of specified interest	User-specified interest areas	Stream-like data
Document Filtering			
Document Categorization	Annotation of documents with meta-level information	Controlled vocabulary	Static databases
Document Tagging			

Classification Objectives

Document routing applications typically have the objective of identifying and selecting documents that potentially satisfy user's interests. Selected documents are presented to the user for review. For example, in the case of the e-mail routing application, the objective is to help the user effectively deal with incoming mail by selecting and classifying relevant e-mail messages automatically.

Document tagging tasks characterize document contents with respect to prescribed subject categories and the names of categories often become an integral part of the document representation, for example, the document index. This recorded meta-information is then used in various ways. For example, the goal of the technical report classification task is to label, that is, tag, technical reports with subject categories that reflect their contents and in that manner support their browsing or retrieval.

Document Categories (Interest Profiles)

In routing applications, the categories (interest profiles) into which the documents are classified generally represent the user's (information consumer's) interests. Thus, the set of profiles is relatively unstable. For example, in a news filtering application, each user specifies categories of news articles that are of interest. These categories may change frequently for an individual user and also vary significantly from user to user.

In tagging applications, the interest profiles are generally subject categories in a specific domain, created by experts in the field. Therefore they are predefined from the perspective of the information consumer. Often these classification categories are referred to as controlled vocabulary. They typically form a hierarchical structure that reflects a domain subject classification. For example, in the classification of technical reports, the categories are technical terms

generally adopted in the technical domain and therefore expected to be known and useful to the user for searching or browsing.

Document Source

In routing applications, the documents often come as a stream of data, presented to the system in unit increments. Typically routing operations are time-sensitive and it is expected that the decision about an individual document is made instantaneously. This does not exclude the possibility of presenting the results to the user in the form that reflects the degree of their relevance to the user's interest.

In tagging applications, the documents are frequently stored in relatively static databases, although it is easy to imagine applications in which document classification via controlled vocabulary needs to be applied to streams of data.

Again, the e-mail application is a typical example of a routing application. The system makes an instantaneous decision about the type of the incoming message and presents it as the most recent relevant message for the given category. In contrast, the main emphasis of the technical reports classification problem is not a time critical processing of individual documents but creating a global structure that captures document inter-relations and makes accessible all the documents in the collection.

Types of Classification Applications

Although quite similar in nature, the specific characteristics of the various document classification applications require use of a wide range of techniques. Two critical areas are relevance measure and classifiaction decision.

Relevance measures - Methods for measuring the degree of match between documents and interest profiltes

Classification decision - Mechanisms for making classification decisions and regulating the quality of classification process using thresholds on document relevance scores

Document Routing

In both routing and filtering applications, users create interest profiles to represent their interests and documents are routed based on their relevance to the expressed interests. Routing is a "data-oriented" view focused on "routing" of

documents to appropriate "destinations". In routing applications, all documents are usually assigned to at least one interest profile.

For example, an email routing application usually assigns each incoming message to only one mail folder. In other routing applications, documents are processed in the batch mode and presented to the user as ranked according to their relevance. Thus, both the relevance scores and the thresholding methods applied to the relevance scores are significant factors in shaping the result list.

Document Filtering

Filtering is a "user-oriented" view in which the emphasis is on 'filtering out' documents that are not interesting to the user. Filtering is often regarded as a task in which documents come into the system as a continuous stream of data and the system has to make an ad hoc decision whether to present the document to the user. Therefore, in filtering, it is the thresholding method that most explicitly affects the quality of the output. The result list is generally a function of the sequence in which documents are presented to the system.

In filtering applications, some documents may not be assigned to an interest profile or a document may be assigned to all profiles. For example, a news filtering service may route a document to all profiles, if all users expressed an interest in its subject matter.

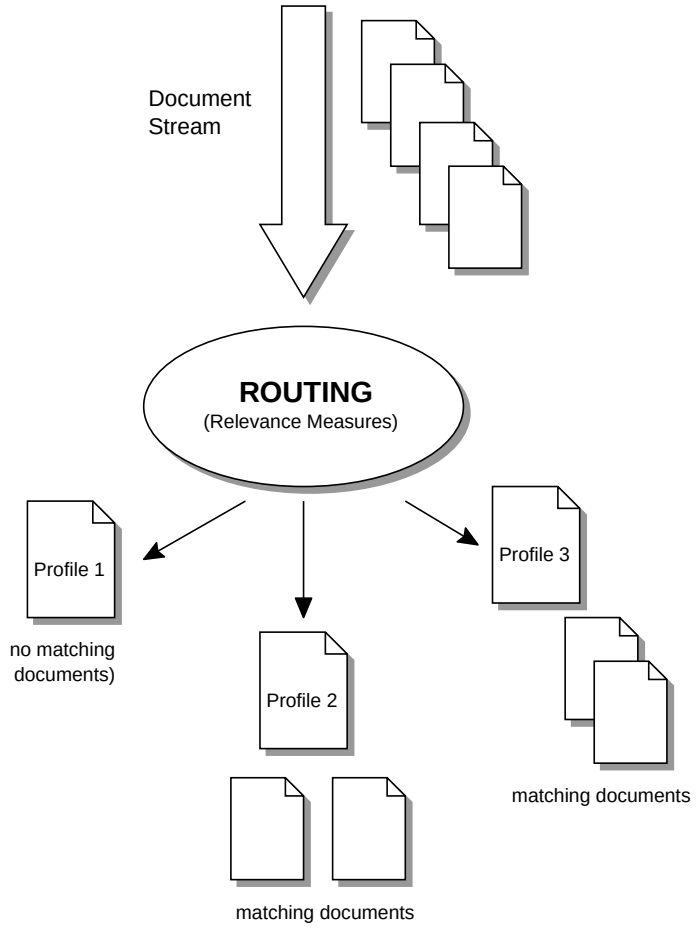

Figure 1-4 Routing Application Workflow

Document Tagging

Document tagging applications focus on meta-characterization of document contents. Document tagging is typically used for data browsing and retrieval based on meta-information about document content (that is, pre-defined subject categories).

Tagging consists of assigning subject titles to documents from a given controlled vocabulary, often of a hierarchical structure, for the purpose of storing and accessing documents. The individual concepts or tags are considered the target interest profiles. For each concept, the system identifies documents in a given collection that should be classified by that concept. When this process is completed, a cross-category analysis can be performed to identify all the concepts to which a particular document has been assigned. In that manner we obtain a

representation of the particular document's content, based on the controlled vocabulary.

In tagging applications, each document is expected to match some, but not all profiles. This assumption enables the threshold to be adjusted based on the number of matching profiles. For example, the program could always assign between 5 and 20 profiles to each document by modifying the threshold to meet this goal.

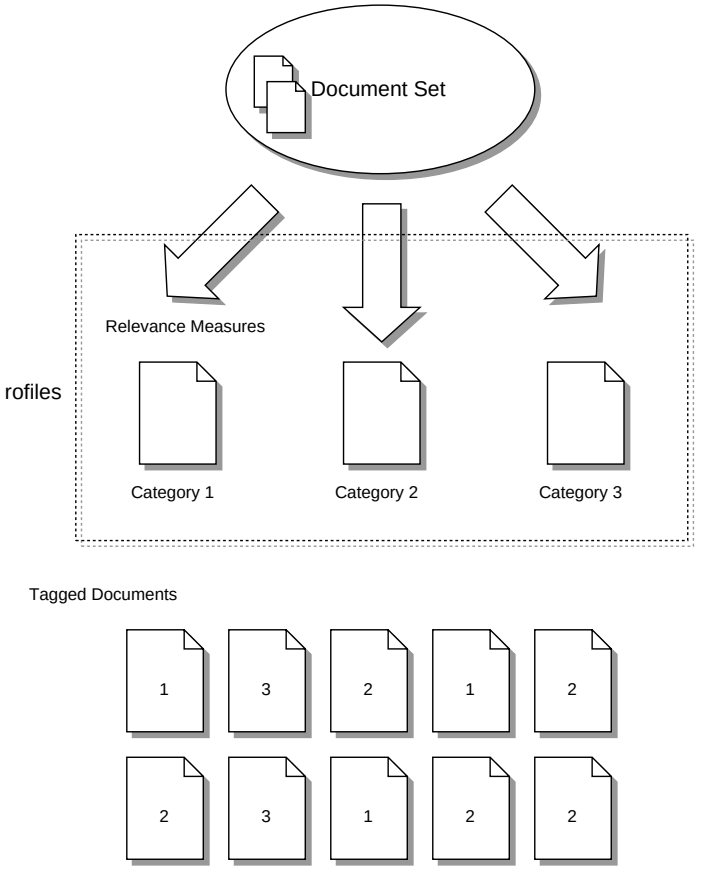

Figure 1-5 Tagging Application Workflow

Writing a Routing Application

This section illustrates the steps in developing a routing application. Because rotuing can cover a variety of applications, you must make key design decisions before choosing the routing policies and functionality you need to use.

Design Decisions

routing or tagging (mark documents or put documents some where)

How are profiles constructed? from queries? users? example documents?

Input: CLARIT Documents or Formatted Text

Will the text you are routing be a CLARIT **Document** or **Corpus**, or will it be raw formatted text? If you have access to CLARIT **Document**s, use a **ProfileSetPolicy** concrete subclass. If you will be routing formatted text, use a **ProfileSetParsePolicy** concrete subclass.

Applications Building Profile Sets

If your application will build its own profile set, use one of the concrete subclasses of **BuildProfileSetPolicy**. Alternately, you may use the **prbuild** program to build an empty profile set.

Select a CLARIT Routing Policy

Based on your application's input (**Document** or a formatted file), select the corresponding policy class from Table "Routing Policy Classes." Figure “Routing Policy Classes” on 6 also shows the available policy classes to choose from.

Routing Policy Classes

CLARIT Document	Text File
LocalProfileSetPolicy	LocalProfileSetParsePolicy
LocalBuildProfileSet	

For example, for local routing applications that route documents in text files (instead of CLARIT documents), initialize the **LocalProfileSetParsePolicy** capability.

Initialization

CLARIT includes a default **main()** function, which prints out CLARIT-specific copyright information and calls the **CCMain()** function. Any exceptions raised by the CLARIT runtime are caught by the default **main()** so that the program exits gracefully. **CCMain()** also initializes threads and ensures the synchronization of the initialization of static variables.

```
int CCMain(int argc, char *argv[])
```

To initialize a CLARIT routing application, you must instantiate **CLARITPolicy**. This automatically initializes the set of available options and handles help and usage messages. The *argc* and *argv* parameters are modified to reflect the fact that CLARIT options have been removed from the list. This allows CLARIT to co-exist with other packages.

```
CLARITPolicy clp(argc, argv, PROGRAM_NAME, options, cachingOptions);
```

The next step is to initialize the desired CLARIT capabilities that you identified in the design step in Section "Design Decisions," on page 1-14. Capabilities can be registered with a **CLARITPolicy** instance (as they are here) so that only one variable must be passed around to other routines and modules.

```
ProfileSetPolicy *psPolicy  LocalProfileSetPolicy::Create(&clp);
```

When you instantiate a subclass and pass it a reference to the global **CLARITPolicy** object, the subclass registers itself with that object under a tag returned by the **GetCapabilityTag** method. Applications can later obtain the concrete instantiation by calling **GetCapability** on the **CLARITPolicy** object, and using the returned tag. Although the concrete subclass is used to instantiate the capability, applications should treat the instance as the abstract superclass once the class is instantiated.

Applications that Do Not Use CCMain

See *Creating CLARIT Applications* for details.

Define Program Options

See *Creating CLARIT Applications* for details.

Creating a ProfileSet Database

A routing or tagging application requires two basic objects: a profile set containing the interest profiles, and the documents to be routed against those profiles. The first step in creating a profile set is creating the empty profile set. Then the profile set is populated with interest profiles.

Create an Empty Profile Set Database

A **ProfileSet** is an indexed database of **InterestProfile**s. An **InterestProfile** can be created, edited, and deleted from the **ProfileSet**. Remote client applications cannot build **ProfileSet**s; however, if using the **RemoteEditProfilePolicy**, profiles can be edited by clients.

You can build an empty profile set database in one of two ways:

- Use **prbuild** to create an empty profile set database.
- Instantiate a **BuildProfileSetPolicy** instance and use its **BuildProfileSet** or **BuildNamedProfileSet** methods

Then use one of the **ResetStatistics** methods to set statistics from a reference database as described in "Managing Statistics," on page 1-18. The database is now ready to be populated with profiles.

Create Interest Profiles

An interest profile must have terms and a relevance threshold. Interest profiles may also have a title, an abstract that contains the user's query text (before terms are identified), a string specifying what to do with documents matching the profile (its destination), and a Boolean constraint.

Specify Terms

There are many ways to create an **InterestProfile**. In most routing applications, interest profiles are created by the user based on the user's knowledge of the topic or with the aid of various knowledge or language resources such as encyclopedias, specialized dictionaries, thesauri, etc. In the CLARIT System, the user's textual description of the topic is processed using CLARIT NLP. CLARIT identifies prominent phrases and uses them to create a topic profile - a vector of phrases from the topic description.

A user can construct queries agains a database that is simliar to the incoming documents. When the user is satisfied with the query results, the query can be converted to an **InterestProfile**

A user can create a profile by specifying an interest using natural language, much like creating a query. Typically, users create quite short descriptions of their topics of interest, ranging from a couple of words to a couple of sentences. CLARIT parses the user's input and identifies the terms. Documents are routed against these terms. Then the user evaluates the routing results, and edits the profile. This approach is more time-consuming that using the query method.

In situations when the user is interested in obtaining more of a certain kind of document and can provide the system with example documents, the profile can be automatically generated for the user. The interest profile can be constructed in part by extracting the thesaural terms from example documents provided by the user.

Another method for creating an **InterestProfile** is to use a training database. A training database is a CLARIT database containing documents that include fields naming the profiles that each document should match. Profiles can be developed automatically using this field information by doing a thesaurus extraction against all the documents assigned to a profile, and using those terms as the profile.

Set Relevance Threshold

In addition to terms, a profile must have a relevance threshold, defining the score a document must have in order to matched to that profile. After specifying a profile's terms, use the **InterestProfile**::**UpdateThreshold** method to calculate its threshold, based on the reference statistics. The threshold must be recalculated any time the reference statistics or the profile's terms change.

Set Term Weights (Importance Coefficients)

Profile terms are assigned weights that reflect the terms' discriminatory value with respect to the relevant documents.

[fix] In absence of the natural term frequency statistics, term importance coefficients play a particularly significant role in the weighting of profile terms. In general, the terminology included in CLARIT profiles can be divided into two main categories: the 'positive' terminology which provides evidence for document relevance, and the 'negative', or distractor terminology, which characterizes non-relevant documents. The term importance coefficients are, in fact, instrumental to enabling the positive terminology to capture relevant documents and negative terminology to deflect non-relevant documents effectively. The positive profile terminology is assigned importance coefficients of positive value while the distractor terminology is assigned negative or zero value

Managing Statistics

The statistics used for classifying documents are crucial to the success for the results. The terms in a profile are weighted by statistics from a reference database. These statistics reflect term frequency and term distribution, ideally in that of the documents to be routed. In the absence of the actual statistics of the documents to be routed, the statistics are determined based on a large set of relevant training data. And, when training data is not available, term weights are approximated by statistics from a general reference database.

The statistics used to calculate relevance scores will vary based on each application's requirements. The following issues should be considered:

- Source of statistics: The nature of the document set used to generate the statistics.
- Amount of information: How large the set of statistics should be for accurate profile matching.
- Application performance requirements: How important processing speed is versus profile matching accuracy.

Documents Used for Statistics: The documents used to generate the statistics should reflect the term usage in the documents being classified. For applications classifying static data that may already exist as a CLARIT **Corpus**, the statistics used for profiling ideally are the statistics from the documents being classified. This is the best case. When classifying a stream of documents, statistics can be generated from a database built from older documents of the type to be classified, such as old e-mail messages. It is hoped that these statistics accurately reflect the terminology used in incoming mail messages. In the worst case, statistics from a general database, such as an encyclopedia, can be used initially. As documents are classified by CLARIT, they can be used to generate reference statistics.

Using documents very similar to those being classified is the most important factor in good classification. Not only should the reference documents use the same terms, but term frequency and distribution should be similar. If you are routing different types of documents, perhaps from different data sources, each should have its own reference statistics.

Amount of Text: In general, the larger the amount of text used to produce the statistics, the more reliable the statistics will be. It is desirable to use at least several megabytes of text, although it is rarely necessary to use more than 100 megabytes of text.

Performance: While a larger reference database will increase the reliability of the statistics, it may slow down the routing processing. If processing speed is a priority, use smaller reference databases. This means even if a very large database is available, for better performance, use a representative subset of the documents.

Maintaining Statistics for Tagging Applications: For applications that analyze streams of data, we recommend that the reference statistic database contain archived data from approximately the previous 30 days. The reference database should contain at least 15 megabyte to 25 megabyte of data. If the data stream is considerably larger than this, only a randomly sampled fraction is needed. Because the data stream will probably change over time, it should be rebuilt approximately every week. This causes the oldest week's data to be discarded, replaced by the latest week's data. That is, every week, the oldest 25% of the reference statistics should be replaced with new documents from the data stream.

Setting a reference database

After an empty profile set is created, you must associate a set of statistics with it. Use one of the methods described in the Table "Statistic Methods."

Statistic Methods

Class	Method	Input
ProfileSet	**ResetStatistics**	**DocumentSet**, **SourceScanner**
ProfileSetPolicy	**ResetStatsFromCorpusName**	**Corpus** name

Updating Statistics

If you understand the nature of the incoming documents, you should base the statistics on a sample of those documents. If the nature of those documents changes, you will need to change or update the statistics you are using. The incoming document stream may change over time, for example, as the focus of news stories changes.

If the nature of the documents to be routed experience a one-time change, you should change the statistics. If the documents experience an evolutionary change, you can periodically change the statistics or add new document to the statitics as the documents are routed. Both of these operations are expensive; try to avoid them if unnecessary.

Resetting Statistics

If you decide it is necessary to use a different set of statistics for a profile set, use one of the reset statistics calls. This is an expensive operation. While this is being done, the database is locked and clients cannot access it. To minimize this disruption, as with other database maintenance, you should gather a group of documents, then periodically, perhaps overnight, add the statistics.

Adding Incoming Documents to Statistics

Incoming documents can be added to the database to more accurately reflect the incoming documents if necessary. Adding statistics from incoming documents involves re-indexing the reference datase. This is an expensive operation. While this is being done, the database is locked and clients cannot access it. To minimize this disruption, as with other database maintenance, you should gather a group of documents, then periodically, perhaps overnight, add the statistics..

Routing Documents

Once a profile set is available, a CLARIT **DocumentSet** or text stream can be routed against it. Routing produces a **MatchedProfiles** instance or a **MatchedDispatch_Iterator**, depending on the method used. To iterate through the documents, use the **StartResultsIteration** method.

Any document classification application uses the same methods for routing. What to do with the documents once assigned to interest profiles differ between routing and categorizing.

Routing Application: Deliver Documents to Their Destination

Tagging Application: Label Documents with Their Category

can store this info elsewhere, perhaps using DocumentSubsets.

Notes on Building and Running a Routing Application

Include Files

When compiling and linking the example CLARIT Routing application, use the following header files:

C Header Files

```
#include <cstdlib>
#include <climits>
#include <cstdio>
```

C++ Header Files

```
#include <iostream.h>
#include <fstream.h>
```

CLARIT Header Files

```
#include <clarit/basics.h>
#include <clarit/clsupp.h>
```

Include the h file corresponding to any classes your application uses.

CLARIT Header Files for Routing Applications

Include the h file corresponding to any classes your application uses.

#include <clarit/profenv.h>

Libraries

On Windows NT, libraries are specified as `cl_`*<libname><build_extensions>*`.lib`. On UNIX, libraries have the form `libcl_`*<libname><build_extensions>*`.a`. When building a CLARIT Routing application, you must link with the following libraries (only the libname poriton is listed):

routing, query, xtract, corpus, nlp, transact, clp, and **ccl**

On Windows NT, you must link against the sp library as well.

For example, NT, you would link with the following libraries for a local applciation:

cl_routingmel.lib cl_querymel.lib cl_xtractmel.lib cl_corpusmel.lib cl_nlpmel.lib cl_transactmel.lib cl_clpmel.lib cl_cclmel.lib cl_sp.lib