Windows Eudora Indexed Search

Last Updated: February 15, 2006

1 Overview

The Indexed Search feature (aka "Ultra-Fast Search" in marketing materials) was added in Eudora 7. The core engine utilized in the Indexed Search feature is X1. The majority of the changes for Eudora 7 were behind the scenes – much of the user interface remained unchanged (see section 6 "Future Directions" for proposed future changes to the interface).

This document attempts to give a brief overview of the Indexed Search implementation with a few specifics in certain areas of interest. Suggestions for future improvements that could be made are also covered.

2 Indexing

Indexing is the process by which the "Search" directory is populated with the data that allows X1 to provide search results with such stunning speed. Indexing is always done in the main thread, usually at idle, because Windows Eudora's current mailstore relies on access occurring in the main thread.

The schema, that is the format and type of data that is stored in the X1 index, is set up in SearchManager::Info::InitX1DB in "SeachManagerInfo.cpp". Unfortunately X1 does not currently provide any mechanism to add a new field to an existing schema without invalidating all users' indexes. In fact, the version of X1 currently used by Eudora does not provide any versioning of the index at all. That is, X1 would not recognize if it was told to load an index with a different schema than that setup in the schema initialization. Eudora provides this capability itself by checking the schema version as stored in "Search.xml" (see section 4 "Indexed Search Data"); if the schema version doesn't match the current schema being used by Eudora, then Eudora erases and regenerates the index. As such, schema changes should only be made when absolutely necessary, to support important new functionality.

2.1 Types of Index Maintenance

2.1.1 Add or Update Scan

The "Add or Update Scan" is a full scan of all mailboxes and messages and is used to either build the initial index or to verify that the index is up to date (verification currently only occurs when initiated by the user via Tools->Options->Find Messages->Index Now). The scan adds any messages that are not indexed and removes and re-adds any messages that are indexed, but are not up to date in the index.

2.1.2 Culling Scan

The "Culling Scan" is a full scan of the index which removes any indexed items for which there is no longer a corresponding message. It currently only occurs after a "Add or Update Scan" is initiated by the user via Tools->Options->Find Messages->Index Now.

2.1.3 Index Update Actions

"Index Update Actions" are queued whenever any change happens in Eudora which should result in an update in the index. Such changes include, but are not limited to, receiving new email, transferring or deleting messages, changing any message attribute, renaming or moving a mailbox, etc. "Index Update Actions" are normally processed at idle, however if a search is initiated any pending index update actions will be processed before the search is performed, which ensures that searches will not return outdated results (e.g. listing messages that have been deleted or that no longer match the search criteria).

3 Searching

3.1 X1 Searching Capabilities

X1 supports:

- Exact word or phrase searches
- Prefix word or phrase searches
- Search result set negation (i.e. get the inverse of the search results - everything not in the search results)
- Combination of search results via AND
- Combination of search results via OR
- Word lengths of up to 39 characters (in theory this can be changed when setting up the X1 database, but in actual practice it was found that doing caused Eudora to crash with the version of X1 we used for Eudora 7.0)

Other X1 search notes:

- Runs of whitespace and/or punctuation are ignored and treated as a single word separator
- X1 caches search results. Therefore slightly changing a search (i.e. changing only one term) and repeating the entire search operates quickly without Eudora having to perform any optimization of its own

3.2 Current Search Process

- Process all search terms
    - Get results for a single search term
    - Get results for next search term (if any) ANDing or ORing with previous results based on whether user has "Match All" or "Match Any" selected
    - Repeat above step for each additional search term
- Find set of messages that match checked mailboxes (see 3.3.1 "Find Messages in Mailboxes" for details) and AND with current results
- If the "Include Deleted IMAP Messages in Results" is turned off, then find the set of messages that are not IMAP deleted and current results
- Add results to search result list
    - For each result, query X1 and retrieve enough information to display search results
    - As results are accumulated periodically process them and add them to the search result list

3.3 Searching Tricks

3.3.1 Find Messages in Mailboxes

When indexing and searching, we prefix all relative paths with "UniqueRootPrefixForEudoraMailboxPaths" to ensure that we can uniquely find mailboxes. For example consider the following directory structure:

```
In.mbx
Foo\Bar\In.mbx
Test\Foo\Bar\In.mbx
```

If we were searching for mailbox path contains the phrase "In.mbx" or "Foo\Bar\" we would incorrectly find the other mailboxes that contain those sub-paths. By instead indexing the above mailboxes as:

```
UniqueRootPrefixForEudoraMailboxPaths In.mbx
UniqueRootPrefixForEudoraMailboxPaths Foo\Bar\In.mbx
UniqueRootPrefixForEudoraMailboxPaths Test\Foo\Bar\In.mbx
```

We can now uniquely find "UniqueRootPrefixForEudoraMailboxPaths In.mbx" and "UniqueRootPrefixForEudoraMailboxPaths Foo\Bar\".
The string "UniqueRootPrefixForEudoraMailboxPaths" was chosen because:

- It is less than 39 characters (the default maximum word length)
- It is very unlikely to appear as an actual IMAP folder name (POP folder names were not an issue for uniqueness because they are suffixed with .fol)

Note that the prefix only affects the *indexed* value of the relative path. X1 allows us to index one value and store another value. For indexing and searching we include the prefix. When retrieving the mailbox relative path from X1 we do not need to worry about removing the prefix because the storage value does not include the prefix.

3.3.2 Finding Numeric Values

For numeric searches X1 only supports searching for exact values. X1 does not provide any built in support for numeric range searches. This was a problem because we wanted to support as much of our classic search capabilities as possible and a lot of Eudora's search locations (Priority, Attachment Count, Junk Score, Date, and Size) require numeric range searches (i.e. "is greater than", "is less than", "is after", "is before").

Luckily numeric range searches can be performed with X1 provided numeric values are encoded in the proper way. There's more than one way this could be done, but the method used for Eudora 7 was encoding the number using a simplified Roman numeral approach (i.e. only additive). In roman numerals M = 1000, C = 100, X = 10, I = 1. When searching for "is greater than" we combine multiple prefix searches to find all possible ways that a value could be greater (one search for each Roman numeral digit). When searching for "is less than" we simply search for "is greater than or equal to" and negate the results.

Example:
4328 is encoded as "MMMM CCC XX IIIIIIII".

To find all numbers > 4328 we look for (unique search strings are bolded):

MMMM && **CCC** && **XX** && **IIIIIIII** OR
MMMM && CCC && **XXX** OR
MMMM && **CCCC** OR
MMMMM

Please note that the complexity of the above search in terms of X1 is 7 unique searches and 3 logical operations.

3.3.2.1 Limiting the Numeric Range of Dates

Eudora supports searching for messages by date or by age (in days). Because both date related searching methods are based on a number of days, we're able to limit the numeric range of dates by indexing the number of days since 1970 (not that far fetched, since we store dates as number of seconds since 1970). Using our approach, X1 can currently support a value of roughly 39,000, which will last until sometime in the year 2076.

4 Indexed Search Data

Indexed Search data is stored in the "Search" folder, which in turn is found in the Eudora folder alongside email and all other special folders. "Search.xml" is the only file in "Search" which is created by Eudora, all the other files are created by X1.

The meaning of the XML tags of "Search.xml" is as follows:

- IndexedSearchData – Root level tag.
- DataFormatVersion – Version of XML data. Not expected to be needed because of the extensible nature of XML, but included just in case anyway.
- SchemaVersion – Version of Eudora's X1 database schema. The version of X1 currently used by Eudora has absolutely no inherent versioning, so Eudora has to provide it on its own. If the schema version as stored in "Search.xml" does not match the current version that Eudora expects when it loads, then all indexed data is erased and Eudora begins a new full scan.
- IndexingScanType – Currently indexing scan type. Scan types:
    - st_None = 0 – No scan
    - st_ReadyForNextScanType = 1 – Current scan finished. Ready to start next scan type when next idled.
    - st_AddOrUpdateItemsScan = 2 – Main scan when items are added or updated.
    - st_CullingScan = 3 – Culling scan. Scans through all X1 items and removes any for which Eudora does not have a corresponding message.
    - st_ProcessingIndexUpdateActions = 4 – Currently processing index updates.
- IndexingNextScanType – Next scan type to perform. See IndexingScanType for a list of valid values.
- IndexingMailbox – Current mailbox being indexed – stored as a relative path starting insides the Eudora data folder.
- IndexingCompletionTime – Last time a full scan was completed. Currently only used to determine whether or not an initial full scan needs to be performed, but could be used in the future to allow full scans to be performed on a schedule.

- SortColumns – Contains a series of "SortBy" subtags, which are passed to X1 when performing a search and control the order in which search results are returned by X1.
  - SortBy - Contains single number representing sort column:
    - BY_STATUS = 1
    - BY_JUNK = 2
    - BY_PRIORITY = 3
    - BY_ATTACHMENT = 4
    - BY_LABEL = 5
    - BY_SENDER = 6
    - BY_DATE = 7
    - BY_SIZE = 8
    - BY_SERVERSTATUS = 9
    - BY_MOOD = 10
    - BY_SUBJECT = 11
    - BY_REVERSE_STATUS = 129
    - BY_REVERSE_JUNK = 130
    - BY_REVERSE_PRIORITY = 131
    - BY_REVERSE_ATTACHMENT =132
    - BY_REVERSE_LABEL = 133
    - BY_REVERSE_SENDER = 134
    - BY_REVERSE_DATE = 135
    - BY_REVERSE_SIZE = 136
    - BY_REVERSE_SERVERSTATUS = 137
    - BY_REVERSE_MOOD = 138
    - BY_REVERSE_SUBJECT = 139
- IndexUpdateActions – Contains 0 or more index update actions as described by "UpdateAction" subtags.
  - UpdateAction – Describes a single index update action as described by:
    - Type – Indicates type of update action. Values are:
      - ua_Invalid = 0 – Indicates not a valid action.
      - ua_RemoveMessage = 1 – Remove the indicated message from the index.
      - ua_AddMessage = 2 – Add the indicated message to the index.
      - ua_UpdateMessage = 3 – Update the indicated message in the index.
      - ua_RemoveMailbox = 4 – Remove all indexed messages in the indicated mailbox from the index.
      - ua_AddMailbox = 5 – Add messages in the indicated mailbox to the index.
      - ua_ReindexMailbox = 6 – First performs same action as ua_RemoveMailbox and then transforms into a ua_AddMailbox.
    - MailboxPath – Relative path to the mailbox in question starting with the Eudora data folder.

- MessageID – Message ID of the message being affected (if any).

5 Classes

SearchManager – Overall manager and gateway to indexed search functionality.
SearchManager::Info – Contains all of the data members of SearchManager, which helps prevent the public SearchManager.h header file from changing as often during development. Implementation handles data related functionality such as loading and saving indexed search data and interacting with X1 to initialize X1 and set up the index format.
SearchManager::XMLParser – Parses the …\Search\Search.xml file which stores the indexed search data.
SearchManager::Searching – Performs much of the details of performing searches with X1.
SearchManager::Utils – Contains miscellaneous search related utility methods.

6 Future Directions

6.1 Search Results to be Virtual Mailbox

Search results are currently displayed in special view that only supports a small subset of the functionality of a mailbox. The decision to implement search results as a separate view was made in an earlier version of Windows Eudora (Mac Eudora has no such separation) and has since been widely viewed as a bad decision. However, implementation schedules have never allowed the time to be dedicated to address this issue. Version 7 was no different – a new much faster search engine was implemented, but the legacy mechanism for viewing search results was maintained in the interest of time.

Some of the major functionality that is missing from the current list of search results:

- No preview pane
- No drag & drop of messages
- Less sorting options
- Alt-Click does not group like items
- Cannot print from search results
- Message->Send Again is not supported

Again, above is only a partial listing of the functionality that is missing in search results. The planned solution for Windows Eudora 7.1 was to reuse the mailbox functionality in place of the current search results. That is, when the "Results" tab is selected a mailbox style view would be displayed. Much of the current mailbox display code should be reused, but not without a decent amount of effort (or we would have made this transition already). The current mailbox display code is too tightly coupled with the current mailbox collection code (Table of Contents – TOCs). One way to approach reusing the current mailbox display code would be to first rework the code to access a collection of messages through an abstraction. Then for normal mailbox display the collection of messages would be implemented using the TOC and for search results the collection of messages would be implemented as a list of search results.

6.2 Speed of Displaying and Live Updating of Search Results

6.2.1 Speed of Displaying Search Results and 'Search as fast as you can type'

6.2.1.1 Why the X1 Desktop Application Shows Results Faster Than Eudora

Most of the search steps outlined in section 3.2 "Current Search Process" are performed extraordinarily fast – so fast in fact that X1's independent desktop searching application (which uses the same core X1 searching engine that Eudora uses – although a more recent version of the engine) allows the user to search as fast as the user can type. That is the X1 desktop searching application updates the search results live with no detectable delay (try a demo of X1 if you can't envision what this is like).

Part of the speed of the X1 desktop searching application is due to the caching that the X1 core engine performs on each search term – when repeating a complex search because the user has modified the search (typed another letter, etc.), typically only a small part of the search really needs to be performed (see 6.2.2 "Priming the Full Search with Partial Searches" for suggestions of how X1's caching can be exploited). However, the primary reason for the speed of the X1 desktop searching application is that it only needs to partially complete the final step (and associated sub-steps) listed in 3.2 "Current Search Process":

- Add results to search result list
    - For each result, query X1 and retrieve enough information to display search results
    - As results are accumulated periodically process them and add them to the search result list

In contrast Eudora completely performs the entire above step before completing the search. The difference between the two approaches is negligible for searches that only return a small number of results, but is significant for searches that return a large number of results (e.g. try searching for "Anywhere contains 'e'" in Eudora).

6.2.1.2 Using a Virtual List to Show Results Faster in Eudora

Eudora completes adding results to the search result list before returning control to the user because search results are currently displayed using the legacy view. In order to avoid the need to process all search results prior to returning control to the user, a "virtual list" technique would need to be implemented. In a "virtual list" technique, Eudora would process enough search results to fill the visible area of the screen plus a little bit more. During idle, Eudora would then repeat the search and retrieve more results increases the cache of results available to show the user. As the user scrolls more results would be retrieved as necessary. The difficulty with this approach is that it would likely require a major rewrite of the list handling mechanism. It could, however, be considered in conjunction with "Search Results to be Virtual Mailbox".

6.2.1.3 Reasons Why We Might Not Want "Search as fast as you can type"

Even assuming the use of a "virtual list" technique to speed up our ability to display a large number of search results, we may not want to support "search as fast as you can type". It's possible that some additional functionality that we may want to support (e.g. grouping sort of messages by subject - something that an ordinary Eudora mailbox can do) may require post-processing of the search results. While that post-processing may be very fast, and may be worth the slight delay that it necessitates for the functionality that it provides, any post-processing is likely to delay results enough that a satisfactory "search as fast as you can type" experience is no longer possible.

6.2.2 Priming the Full Search with Partial Searches

Any given full search is made up of numerous smaller searches that are combined via X1's AND and OR operators. As mentioned previously, X1 caches each set of search results. This caching is noticeable when the user clicks the "Search" button twice in a row. When a completely new search is first performed, there is a slight delay before the search results are displayed. Repeating the exact same search (or even a slightly modified search) returns results more quickly with very little or no delay (assuming a reasonably small number of search results).

Eudora could avoid the slight delay that occurs when a search is first performed by "priming" sub-portions of the search at key moments:

- When focus leaves the "Mailboxes" tree in the "Mailboxes" tab perform the search for what messages match the currently checked mailboxes
- When the focus leaves an edit field perform that search term
- When the user clicks the "More" button to add a search term perform any previous search terms

"Priming" a sub-portion of the complete search will occur quickly and will likely not be noticeable to the user because:

- The slowest portion of searching – displaying results to the user is not being done
- The sub-portion is a small chunk of the complete search and thus only takes a small portion of the total time needed to perform the complete search

Then when the user clicks the "Search" button the complete search would be repeated (both because X1's API does not allow results to be stored and to ensure that nothing was missed during the "priming" process), but most (if not all) of the search would have already been "primed". Therefore the search will complete very quickly because X1 would already have the necessary results cached.

6.2.3 Live Updating of Search Results

Currently searches are performed only when the user first initiates the search via the "Search" button. Some effort is made to update some of the columns in the search results when messages are transferred, etc. but no effort is made to verify that the messages listed still fit the search criteria. As new messages are received, deleted, or modified the search results will likely no longer reflect the current state of the mail store.

Live updating of search results would occur at idle. In order to use as little time as possible so that the user is not interrupted from doing other tasks, "Using a Virtual List to Show Results Faster in Eudora" would need to be implemented first (otherwise searches with a lot of results would not complete quickly enough). Additionally it might prove necessary to perform small chunks of a search (similar to the priming described in 6.2.2 "Priming the Full Search with Partial Searches") in multiple idle processing slices before re-performing the complete search to ensure that the complete search doesn't take too long (by priming each small chunk it ensures that X1 will perform the complete search very quickly because of X1's caching).

An option to allow live updating of search results to be temporarily turned off might be desired. If so, one logical place for this option would be in the menu of the search setting button (to the right of the "More" and "Fewer" buttons).

6.3 Persistent and Multiple Searches

6.3.1 Saved Searches

Provide a mechanism for saving and loading past searches so that the user can repeat past searches more easily without needing to retype. This could be done via a combo-box menu button, much like the search settings button (to the right of the "More" and "Fewer" buttons). Such a menu could list a brief summary of recent searches.

A way to delete recent search items should also be provided, such as allowing the user to type the "Delete" key while a search menu item is selected. Another possibility would be to provide a special menu item (e.g. "Manage…"), which would bring up a dialog box that could allow the user to:

- Delete recent search items
- Rename recent search items (to a meaningful name rather than the default name)
- Reorder recent search items

6.3.2 Multiple Searches

Windows Eudora currently limits the user to one search window at a time. Although "Saved Searches" would reduce some of the urgency, support for multiple search windows would still be very useful, particularly with live updating of search results. Multiple search windows were not implemented for Eudora 7 for legacy and schedule reasons. If the Search Results window is reworked to support some or all of the above suggested improvements, it would make a lot of sense to add support for multiple search windows at the same time.

6.3.3 Search Mailboxes

"Search Mailboxes" would be special saved searches that would be listed in the mailbox tree, but rather than actually physically containing messages, they would instead list messages that match a search which would occur periodically at idle, updating live behind the scenes. With search mailboxes, the same set of messages could be viewed in multiple ways.

For example, assume a user likes to organize her email by topic and therefore she filters (or transfers) her messages into multiple messages by project, as in:
In
Project1
Project2
Project3
… and so on …

Messages that she wants to track by another criteria, like:

- Messages from her boss

- Unread messages
- Messages that need to be acted on which are marked with a custom “In Progress” label

could therefore be found in multiple mailboxes. With Eudora 7 she has two choices if she wants to also track these messages by other criteria. She can transfer a copy of the messages into other mailboxes (thus taking up extra space) or she can search repeatedly for these criteria. “Search Mailboxes” combines the power of “Saved Searches” and “Multiple Searches” and allows her to have saved search mailboxes that find messages by any criteria, such as the ones listed above.

The details of search mailboxes could be stored in .sbx (search mailbox box – ala .mbx for mailbox) files that are stored as XML that describes the search.

7 X1 Contacts

Business Contact:
Lew Roth <lew@x1.com>

VP of Engineering:
Dejan Nenov <dejan@x1.com>

Engineers:
Eric Schmidt <eric@codeicon.com>
Graham Neumann <graham-neumann@shaw.ca>