Section 11 Archive Search Functions using Email

This section is an introduction to the LISTSERV archive search functions using email, and it is intended to be a reference document for general users with little or no knowledge of database search systems. It does not contain any technical information that general users would have to worry about.

This section will discuss the syntax and operational characteristics of the LISTSERV database subsystem. It is assumed that the reader is familiar with his or her email client and familiar with sending commands to a LISTSERV server.

If you just need a "quick start", read the next two sections for basic instructions. If you want a detailed tutorial on how to use the SEARCH command itself, you might want to skim the next two sections, and then start reading Section 1.3 The SEARCH Command.

11.1 Basic Search Session

To search for the term "Digest=" in the EASE-HOME list on HOME.EASE.LSOFT.COM, create a new mail message addressed to LISTSERV@HOME.EASE.LSOFT.COM and in the body (not the subject) of the message, simply type:

```
Search 'Digest=' in EASE-HOME
```

LISTSERV might respond to you with the following:

```
>Search 'Digest=' in EASE-HOME
-> 10 matches
Item#       Date        Time     Recs     Subject

000058      96/01/26    14:44    41       What happened
000059      96/01/26    18:14    38       Re: What happened
000066      96/02/02    22:51    31       Digest Problem
000074      96/02/03    15:01    75       Re: Digest Problem
000075      96/02/03    18:52    49       Re: Digest Problem
000076      96/02/03    16:27    52       Re: Digest Problem


To order a copy of these posting, send the following command:

    GETPOST EASE-HOME 58-59 66 74-76

>>> Item #58 (26 Jan. 1996 14:44)  -  What happened
      I never touched the Limits= command or the notebook=
All I did was try and add Digest = Yes, Daily
                              ^^^^^^^

I have tried this several times with the same reply message:

>>> Item #59 (26 Jan. 1996 18:14)  -  Re: What happened
 >   I never touched the Limits= command or the notebook=
All I did was try and add Digest= Yes,Daily

                              ^^^^^^^
```

Note: LISTSERV includes excerpts from the indexed postings showing the context of the search term(s). We've deleted all but the first 2 in the example above to save space.

Next, use the GETPOST command to order the specific posts you wanted to read. For instance, if you want to read posts numbered 66 and 74 through 76. You would make another new message (or reply to the response from LISTSERV without quoting the text) and type in the body:

```
GETPOST EASE-HOME 66 74-76
```

LISTSERV would then respond with the desired postings. For the non-VM servers, GETPOST is analogous to the old database command "PRINT". There is no corresponding command for the old database command INDEX, since the response to a SEARCH command includes the index of matching postings.

11.2 Narrowing the Search

It is possible to add further parameters to your search in order to narrow it. You can limit a search by date with a "since. . . " predicate. Likewise, you can limit by sender and/or by the subject line with a "where . . ." predicate. For instance:

Search 'Digest=' in LSTOWN-L since 94/01/01
Search 'Digest=' in LSTOWN-L where sender contains 'Thomas'
Search * in LSTOWN-L where sender is ERIC@SEARN
Search * in LSTOWN-L since 94/01/01 where subject contains 'Digest'

are all valid search commands that will (depending on how well you've crafted your predicate) dramatically reduce the number of entries returned to you.

11.3 The SEARCH Command

This section will introduce the formal syntax of the SEARCH command.

Note: The minimum abbreviation of this command is "S".)

The syntax of this command is a bit complex, and will be introduced step by step.

11.3.1 Basic Search Function

The two most important things you have to indicate when you search list archives are:

1. The name of the list whose archives you want to search.
2. What you want to search the individual documents for.

The name of the list to be searched is specified after the words or phrases to be sought and is prefixed with an IN keyword. For example, we might do this:

```
Search Rosemary in MOVIES
```

This would select all the entries from list "MOVIES" containing the string "ROSEMARY".

Now if you just wanted to see the list of all the movies you can see, you could have used an asterisk as search argument to select all the entries in the list:

```
Search * in MOVIES
```

Note: The mailing list name doesn't have to be in uppercase.

If you want to "narrow" your previous search, i.e. perform additional tests on the documents that have been previously selected, you must omit the IN keyword. In that case, the search will be applied to the previous "hits" and will create a new "hit list".

But in most cases, we will want to search for something longer than one word, for example part of a "key" sentence.

```
Search Hardware problem with a 4381 in IBMFORUM
```

Another problem is that we might not remember the exact original sentence. This is not very important, since LISTSERV will search each word individually: in the above example, any entry that contained the words "hardware", "problem", "with", "a" and "4381" would have matched the search, even if the words appeared in a different order.

But, what if the original document had "4381-13" in it, instead of "4381"? This is again no problem, as LISTSERV does not require the word to be surrounded by blanks to find a match. Case is also ignored when performing the search operation. That is, "problem" would have found a match on "problems"... and "with" would have found a match on "without" or "withstand"! This may sound like inconsistent behavior, but you should keep in mind that it is always possible to "narrow down" a search operation. However, once a document has been excluded from the list of "hits", it is very difficult to bring it back.

Now what if I want to search for an exact string? For example, I am interested in the string "in C". It is very likely that just any document in the database will contain both a "in" and the letter C. But what I am interested in is things which have been written, or programmed, or implemented, "in C". In that case, it is possible to force LISTSERV to group words together by quoting them, as in:

```
Search 'in C' in UTILITY
```

This method can also be used to insert extra blanks between or before words: leading and trailing blanks are normally removed automatically, but they are preserved inside quoted strings. Please note that quotes must be doubled when specified inside quoted strings, as in:

```
Search 'Rosemary''s baby' in MOVIES
```

The search for 'in C' resulted in over fifty hits, because a match was erroneously found against "in clear", "in core", etc. However, I do not want to search for 'in C ' because there might be hits with "in C." or "in C," in the database and I don't want to miss them. If the search respected the capital C, it would no longer find all those irrelevant hits. To do this, enclose your search string in double-quotes instead of single quotes, for example:

```
Search "in C" in UTILITY
```

Note: Single quotes should not be doubled inside double-quoted strings, and vice-versa. Only quotes of the same type as the string should be doubled.

It is important to understand the difference between the two types of quoting. If you request a search for 'TEXT', you will find a match on "TEXT", "Text", "text" or even "teXt". This is the same behavior as unquoted text. However, if you request a search for "TEXT", it will only find a match on "TEXT", not on "text" or "Text".

Quoting is also the only way to search for a reserved keyword like "IN": if you tried "Search in in UTILITY", LISTSERV would report that database "IN" does not exist and would reject the command. This is because the keyword IN indicates the end of your search arguments. If you quote it, however, it will not be recognized and will be searched as you wanted it done. Similarly, if you want to search for an asterisk, you will have to quote it since

```
Search *
```

indicates that all entries should be selected.

Now the problem is that there may be sentences starting with a capital I, e.g. "In C, it would be coded this way:". How can I catch these sentences? Actually, you have been using "complex search expressions" from the beginning without even being aware of it. When you specified a search on

```
Hardware problem with a 4381
```

you had, in fact, been asking LISTSERV for: "Hardware NEAR problem NEAR with NEAR a NEAR 4381". The "NEAR" is implicit, but it may be overridden.

Note: This is a change from the way the "locate" clause was implemented in 1.8b and earlier. Earlier versions used a default of "AND" instead of "NEAR" between discrete search terms. The difference is that

```
Search JOE SMITH in XYZ-L
```

looks for JOE and SMITH close to each other rather than simply looking for instances of both terms in the entire document. You can still use AND explicitly. Note that 'a NEAR b NEAR c' is defined as '(a NEAR b) AND (b NEAR c)', so the NEAR operator is not fully commutative.

You may even use parenthesis if needed:

```
Search ("in C" or "In C") and program in UTILITY
```

The "NEAR" can still be implied, as in:

```
Search wooden chair (blue or green) in CHAIRS
Search (wooden chair) or (plastic chair) in CHAIRS
Search plastic chair (blue or green but not streaked) in CHAIRS
The following commands are strictly equivalent:
Search (wooden chair) or (plastic chair not blue) in CHAIRS
Search chair (wooden or (plastic not blue)) in CHAIRS
Search chair (wooden or (plastic but not blue)) in CHAIRS
Search chair NEAR (wooden OR (plastic AND NOT blue)) in CHAIRS
```

11.3.2 Date Specifications

Since each document has been assigned a "date/time" field, it is possible to select documents based on this date field. This is accomplished by appending "date search rules" to the search expression, as in:

```
Search problem (serious or severe) in BBOARD since july
Search problem in BBOARD since oct 85
Search symptom in BBOARD since 12/28
Search error report from 12 january to august in BBOARD
Search user complaint until 18 sept in BBOARD
Search data check since today 11:53 in EREP
```

The default values for omitted arguments are always chosen so as to exclude as few entries as possible. For example, "July" would mean "1 July 00:00:00" in a SINCE specification, and "31 July 23:59:59" in an UNTIL clause. The only exception is the year field, which always defaults to the current year.

11.3.3 Keyword Search Specifications

The last thing you may wish to search is the "keywords" list. For example, you might want to select those plastic chairs which cost less than 50 dollars. It is assumed that the price will vary often (maybe almost daily), and that it is therefore kept externally from the document describing the chair. Thus, you would have a "Price" keyword which you could search in the following way:

```
Search plastic chair in CHAIRS where price < 50
```

You may of course use complex expressions (with parenthesis) in the WHERE clause. There are new comparison operators available for this clause, like IS, CONTAINS, all the usual arithmetical comparison operators, and some more. However, the AND operation is no longer implied, but it can still be specified explicitly of course:

```
Search plastic chair in CHAIRS where price < 50 and avail > 4
```

The problem now is that, as the search commands become more and more complex, they will no longer fit in a single line. To solve this problem, we begin the command with the string "// " (two front-slashes and a space) and follow it with the SEARCH command and the search specifications. Any database command ending in a comma indicates that more is to follow on the next line. This process can be repeated several times if desired.

```
 // Search chair (wooden or (blue or green but not streaked)) ,
       in CHAIRS ,
       where price < 50 & avail > 4

// Search chair (wooden or (blue or green but not streaked)) ,
 in CHAIRS where price < 50 & avail > 4

// Search chair (wooden or ( ,
  blue or green but not streaked) ,
  ) ,
  in CHAIRS where price < 50 & avail > 4
```

The only "trick" about this continuation line business is that you should always keep quoted strings on a single line. The process of identifying continuation lines and concatenating them afterwards may cause unwanted blanks to be inserted in the command line, which is no problem outside a quoted string since blanks are ignored, but might cause erroneous results in a quoted string.

If you want to search for several possible values in a given keyword, you do not have to repeat the keyword name and operator:

```
 // Search * in BBOARD where ,
  subject contains (PC or (Personal and coputer))
is strictly equivalent to:
// Search * in BBOARD where ,
subject contains PC or ,
(subject contains Personal and subject contains computer)
```

However, it should be noted that this "factorization" is performed according to the rules of logic, which may not necessarily match those of English grammar. This removes any possible ambiguity as to the meaning of these clauses. Let's consider the following:

```
machine does not contain (IBM and DEC)
```

This clause will get translated into:

```
machine does not contain IBM and machine does not contain DEC
```

In English, you would probably say "machine contains neither IBM nor DEC". This is how LISTSERV will understand it. However, if you read the clause aloud, you will probably not pronounce the parenthesis and will end up saying "machine does not contain IBM and DEC"; in other words, "machine does not contain both IBM and DEC", which is a totally different thing. The "English meaning" could be obtained with the following clause:

```
not (machine contains (IBM and DEC))
```

In the former case, the negative "does not contain" operator is inserted inside the parenthesis. In the latter, only "contains" is moved, and the negation remains outside.

```
// Search gateway problem ,
  in BBOARD ,
  since sept 86 ,
  where sender contains (john or paul but not mick) ,
  and subject does not contain lost
-> 5 matches.
Item #   Date   Time  Recs   Subject
------   ----   ----  ----   -------
000012 87/10/18 13:09   12  The gateway has stopped working
000017 87/08/24 09:18    9  Glory glory alleluja! Again!!!
000018 87/10/18 13:09    8  You know what? It WORKS!!!
000024 87/10/18 13:09    7  Guess what happened today?
000205 87/10/04  16:59   9  Who's going to babysit it today?
You  might now wish to narrow your search down to exclude
postings whose subject contains "work".  For instance,
// Search gateway problem ,
   in BBOARD ,
   since sept 86 ,
   where sender contains (john or paul but not mick) ,
   and subject does not contain (lost or work)
-> 3 matches.
Item #   Date   Time  Recs   Subject
------   ----   ----  ----   -------
000017 87/08/24 09:18    9   Glory glory alleluja! Again!!!
000024 87/10/18 13:09    7   Guess what happened today?
000205 87/10/04 16:59    9   Who's going to babysit it today?
```

11.3.4 Phonetic Search

There may be cases where you are looking for a certain value of a keyword, the exact spelling of which you cannot remember. In these cases, it may be useful to try a phonetic search. A phonetic search will yield a match for anything that "sounds like" your search string, as dictated by a predefined algorithm which is of course not perfect. It may give a hit for something which does not actually sound like your search string, or, more rarely, omit a keyword which did sound like what you entered. The main reasons for this are that the algorithm must be fast to execute on the machine and therefore not too sophisticated, and that the way a given word is pronounced depends on the idiom in which the word was written. For example, the phonetical transcription of the name "Landau" will be different in French, English, German and Russian. Thus, it is impossible to decide whether a word sounds like another if the language in which the words are pronounced is not known (and of course LISTSERV does not have any way to know it).

Phonetic searches are performed through the use of the SOUNDS LIKE and DOES NOT SOUND LIKE operators, which are syntactically similar to CONTAINS and DOES NOT CONTAIN. That is, you could do something like:

```
Search * in PHONEBOOK where NAME sounds like WOLF
```

There is a little trick with the SOUNDS LIKE operator that you should be aware of. If your search string (WOLF in our above example) is a single word, it will be compared individually to all the words in the reference string (i.e. the data from the database), and will be considered a hit if it "sounds like" any of the words in the reference string. Thus, the search word "Ekohl" sounds like the reference string "Ecole Normale Superieure" because it matches the first word. If the search string contains more than one word, the search and reference strings will be compared phonetically as a whole (and "Ekohl Dzentrahll" will therefore not match "Ecole Normale Superieure"). Note that any search string containing more than a single word must be quoted, as explained in the previous sections.

```
 > Search * in BITEARN where site sounds like (COHRNEAL and
LAPORRADRY)
-> 3 matches.
Ref# Conn  Nodeid   Site name
---- ----  ------   ---------
0292 87/03 CRNLASSP Cornell University Cornell Laboratory of Atomic
0301 87/03 CRNLION  Cornell University Cornell Laboratory of Plasma
0307 87/06 CRNLNUC  Cornell University Laboratory of Nuclear Studes

> Search * in BITEARN where SITE sounds like HOPTIKK
-> 2 matches.
Ref# Conn Nodeid Site name
 ---- ---- ------ ---------
 0751 87/09 FRIHAP31 Assistance Publique - Hopitaux de Paris
 2120 87/04 UOROPT University of Rochester The Institute of Optics

 > Search * in BITEARN where SITE sounds like SCHIKAGO
 -> 1 match.
 Ref# Conn Nodeid Site name
 ---- ---- ------ ---------
0140 86/03 BMLSCK11 Studiecentrum voor Kernenergie (SCK/CEN), Mol,
```

Above, the first command shows an example of accurate phonetic match, where the result is exactly what the user expected. In the second example, the user found what he was looking for ("Optics"), but an additional unwanted entry was selected. This is by far the most common case. The last command is a typical example of phonetic clash, where the algorithm did not translate the search string into phonetics as the user expected it, with the result that the desired name ("Chicago") was not found and that completely irrelevant entries were presented instead.

The phonetic matching algorithm used by LISTSERV is a slightly modified version of SOUNDEX -- a well-known algorithm that provides reasonably accurate matches at a very low CPU cost. Although it gives best results with the English language, for which it was originally designed, it is not too strongly tied to it and can still be used with other languages. It is of course absolutely impossible to write an program that would work for all the languages in the world, or even for the most widely used ones, since their interpretation of the most common combinations of letters are completely incompatible.

11.3.5 What to do about “100 matches (more available)”

LISTSERV limits the number of matching records in a given response to no more than 100. This is done primarily to stop hackers from tying up the server with SEARCH requests on lists with thousands of archived postings, but it also keeps the size of the response down to a manageable level. For instance, sending a "SEARCH *" command for an old, very large list could result in a response measuring in megabytes if not for the 100-record limitation.

There are a couple of different approaches to a solution:

- (Preferred) Narrow your search parameters as explained in Section 1.2 Narrowing the Search and following.
- Specify the first record for your search so that LISTSERV knows to start in a certain place. For instance, if you sent a search command like

```
  search * in lstown-l where sender contains nathan@example.com
```

  that resulted in more than 100 "hits", and the last four hits were something like

```
007696 95/08/23 16:02   13   Re: How to send 'urgent' messages
to digest users?
007698 95/08/23 18:10   52   Re: Blocking expletives
007699 95/08/23 20:00   41   Re: How to send 'urgent' messages
to digest users?
007716 95/08/25 10:34   33   Re: ignore subsequent lines to
listserv?
```

  you could send a followup search command using the following syntax:

```
search * in lstown-l.7716- where sender contains nathan@example.com
```

  to get the next 100 hits starting with message number 7716. Note that it is important to include the trailing hyphen after the starting message number, as otherwise you will get back a response containing only a reference to the message number you specified.

11.3.6 Specifying the Last “*n*” Posts as a Range

Starting with LISTSERV 1.8d it is possible to specify that your search criteria be applied to only the last n posts in the archive. For instance, say you are only interested in checking the last 50 postings of LSTOWN-L to see if nathan@lsoft.com posted. You would send

```
search * in lstown-l.last50- where sender contains nathan@lsoft.com
```

Again you would have to use the hyphen after the number. The number n can be any integer, but as with any other search, if LISTSERV finds more than 100 hits only the first 100 will be returned to you. Thus it would be possible for n to be 1000, or even 10,000, but you will still have to narrow your search if you want more hits.

11.3.7 Exact Syntax Description

This section describes the exact syntax of the "SEARCH" command in technical terms.

11.3.7.1 General Syntax

`Search`	`search-rules <optional-rules>`
	
	`Optional rules are:`
	`date-rules`
	`keyword-rules`

The optional "date-rules" and "keyword-rules" arguments may appear in any order.

Date rules specification

You may optionally restrict the search to only those entries that lay within a given interval of time. This is accomplished by specifying one of the following date rules:

```
SINCE date-spec <time-spec>
FROM date-spec1 <time-spec1> TO date-spec2 <time-spec2>
UNTIL date-spec <time-spec>
```

The format of a "date-spec" is quite complex because of the number of different ways date/time specifications are usually expressed:

```
TODAY
yy
dd mm
<dd><->monthname<-><yy>
mm/yy
mm-yy
yy/mm/dd
yy-mm-dd
yyyymmdd
```

Month names can be abbreviated to any length. If there is an ambiguity, the first month in chronological order is retained. For example, "J" would mean "January", "JU" would be "June" and "JUL" would unambiguously select "July".

The format of a "time-spec" is simply <hh:mm<:ss>>.

Note: Case is irrelevant in date specifications. The keywords (SINCE, UNTIL, etc.) have been capitalized only for better legibility and be entered in lower case, if desired.

11.3.7.2 Keyword Rules Specification

You may request the actual document search to take place only for those entries which match a set of "keyword comparison" rules. The syntax is the following:

```
WHERE kwd-expression
WITH kwd-expression
```

"kwd-expression" is, generally speaking, an mathematical expression of keyword/value comparisons, possibly bound by logical operators. Comparison operators have a higher precedence than logical operators, that is, "A>10 AND B=20" is interpreted as "(A>10) AND (B=20)". The available comparison operators are listed below. All the operators appearing on a given line are synonyms.

```
= IS
^= <> IS NOT
>
<
>=
<=
SOUNDS LIKE
DOES NOT SOUND LIKE
CONTAINS
DOES NOT CONTAIN
```

All these operators are self-explanatory, except the last two which allow you to search the keyword value for a given "substring". That is, "Sender contains jeff" would be true if the value of the "Sender" keyword was "Jeff Smith" or "Jeffrey Donaldson". The case is ignored during the comparison unless the search operand is double-quoted.

If no valid comparison operator is specified between two arguments, "IS" (identity) is assumed. The available logical operators are:

```
^ NOT
& AND BUT
| / OR
```

Note: The logical operators AND and OR have equal precedence and are evaluated left to right.

Finally, keywords and operators can be "factorized" when the same comparison is to be applied to a given keyword and a series of commands. For example, you might enter:

```
    Search * where sender contains ('CS Dept' and (Jack or Phil))
```

This is internally expanded to:

```
// SEARCH * WHERE sender CONTAINS 'CS Dept' AND ,
   (sender CONTAINS Jack OR sender CONTAINS Phil)
```

The expression must always be enclosed in parenthesis, even if it is a simple one:

```
Search * where sender contains (Joe or Morris)
```

This stems from the fact that comparison operators have a higher priority than logical (boolean) ones.

```
WHERE Sender is “Arthur Dent” ,
and Subject does not contain tea

WITH Refcode 8467272 and Location Roubaix

WITH (QTY > 100 | PRICE > 1000) $ MAT = COPPER

Where Sender is (Atiaran@Land or Elena@Land) ,
and Subject contains (‘Be true’ but not Ur-Land)
```

11.3.7.3 Search Rules Specification

Finally, you must specify what is to be searched inside the document. If you do not want anything to be sought at all (e.g. if you are only selecting known items from the database), you can specify an asterisk as a placeholder to waive the search. Otherwise you must specify a mathematical expression where arguments are search strings, possibly bound by logical operators (see Figure 10 for a comprehensive list). The default operator is AND, so that a search for "INTERPRET STEM PROBLEM" will select all entries where "INTERPRET", "STEM" and "PROBLEM" can be found (not necessarily in the same line).

```
Search *
Search ‘I/O’ Error
// Search Interpret (performance or tips, but not
(bug or question))
```

11.3.7.4 Reserved Words and Quoting

When to quote strings

Keyword names and search arguments need not be quoted, unless:

- They are formed of more than one word (search arguments only).
- They contain leading or trailing blanks (search arguments only).
- Their name matches one of the "reserved keywords" of the LISTSERV database system, and appears in a context where it can be mistaken for such. The "reserved keywords" are: FROM, IN, SINCE, TO, UNTIL, WHERE, WITH.
- They contain a parenthesis, logical operator or comparison operator symbol. More generally, you should quote any string that contains one of the following characters:

```
( ) < > = | & ^ /
```

Any non-quoted word will be stripped of leading and trailing blanks and converted to uppercase before the search.

Single-quoted strings

Strings quoted in single-quotes (') are converted to upper case and cause case to be ignored during the search. That is, they behave in the same manner as un-quoted strings as far as the search algorithm is concerned. As a rule of thumb, any string can be single-quoted if desired, even if it does not have to.

Single quotes must be doubled inside single-quoted strings, but double quotes should not:

```
Search '"T''amo, ripetilo, si caro accento' in OPERA
```

Double-quoted strings

Strings quoted in double-quotes (") are not converted to upper case. They result in a case-sensitive search, which means that you should never double-quote a string unless you want case to be respected during the search.

Double quotes must be doubled inside double-quoted strings, but single quotes should not:

```
Search """T'amo, ripetilo, si caro accento" in OPERA
```