OL Search: The On Location™ API 1.1

This release contains files that implement the search engine technology of the On Technology, Inc. product On Location™. The technology in the On Location™ Developer's Kit cannot be incorporated into a shipping version of your product until your company and ON Technology have agreed upon mutually acceptable technical and financial terms.

Description:

The On Location Developer's Kit is a tool for programmers that allows access to the search engine technology of On Location. The kit consists of a library (plus code resource), C and Pascal header files, example XCMD source, pre-built XCMDs and an example stack. This will allow queries on existing On Location indexes from within other applications (including DAs, MPW tools and HyperCard stacks). The kit retains the speed of the On Location product.

Files:

The files included in this release of the On Location API are as follows:

ol_search.h - MPW 3.2 C header file
ol_search.p - MPW 3.2 Pascal header file
ol_search_glue.o - MPW 3.2 library file
ol_search_code - Resource file containing the ol_search engine

Examples files:
ol_search_XFCN.c - MPW 3.2 C source file for example XFCN's
OLSearch_XFCN - Pre-built HyperCard 2.0 XFCN's
OLSearch - Example HyperCard 2.0 stack

API Overview:

Two data types and the related enumerated values are used:

```
enum {
        no_err = 0,
        no_code_resource,
        bad_index,
        damaged_index,
        bad_query_type,
        bad_query_string,
        bad_result_handle,
        result_overflow,    /* non-fatal */
};
typedef short ol_error;

enum {
        filename_matches_exactly = 0,
        filename_starts_with,
        filename_ends_with,
        filename_contains,
        text_matches_exactly,
        text_matches_root_of,
};
typedef short ol_query;
```

Three functions are used to access the search engine:

ol_init:

```
Handle ol_init();                    /* C declaration */
FUNCTION ol_init: HANDLE; C;{ Pascal declaration }
```

ol_init returns a relocatable, non-purgeable handle to the persistent state of the engine. This data is on the order of the size of the “OL File Kinds” file (about 15K). Call this function once upon entry to your code to initialize the On Location search engine. The engine will still perform the query if ol_init is not called provided a zero is passed as the ol_state parameter. If no state is provided then only “generic” file kinds data will be available in the result data (i.e. “document”, “folder”, or “application”).

Note:

If you are running the On Location INIT on the host machine and background indexing is enabled ol_init will turn off background indexing. This is necessary to prevent conflicts.

ol_search:

```
ol_error ol_search(     Handle ol_state,
                        const Str255 index,
                        const Str255 query,
                        ol_query type,
                        Handle result,
                        long *num_hits );

FUNCTION ol_search(     ol_state:    HANDLE;
                        index:       Str255;
                        query:       Str255;
                        type: ol_query;
                        result:      HANDLE;
                        VAR num_hits: LONGINT): ol_error; C;
```

Parameters:

ol_state:	the handle returned from ol_init (or zero, see above).
index:	full path to the index file.
query:	query string. NOTE: this is actually restricted to 100 characters.
type:	one of the values from the ol_query enum.
result:	a handle of text (not more than 30,000 bytes)
num_hits:	actual number of hits of the query (not he number returned in the result).

Result data:

The format of the result data is described by the constants:

```
#define FIELD_DELIMITER     '\t'
#define LINE_DELIMITER      '\n'
#define END_OF_HANDLE       '\0'
```

If the result data grows beyond 30,000 bytes it will be truncated. (This is a limitation of HyperCard fields). Only complete lines will be written. In the case of truncation, result_overflow will be returned from ol_search. num_hits will contain the actual number of hits not the number of lines in the handle. The result handle will be resized to the actual size of the data (including the END_OF_HANDLE character).

Note:

The size of the result handle (less one for the END_OF_HANDLE character) should be used to determine the amount of data to parse. The END_OF_HANDLE character is added for compatibility with HyperCard which requires a null terminated string to be displayed in a field.

Be advised that null characters (currently the END_OF_HANDLE character too) can appear elsewhere in the result handle as well because TYPEs, CREATORs, and FILENAMEs can include them. This may result in the truncation of the results field prematurely when displayed in HyperCard (like in the OLSearch example stack).

If the number of hits (num_hits) is zero the only character in the handle will be the END_OF_HANDLE delimiter.

Return values:

If ol_search encounters an error with the parameters, one of the following values of ol_error may be returned:

no_err:	No problems encountered.
no_code_resource:	The ol_search code resource is not available. Check your build procedure (described below).
bad_index:	The file passed in as the index to be queried was not a valid On Location index.
damaged_index:	The index provided is damaged. You will need to delete that index and reindex the target volume.

bad_query_type: Value given for query was not in the range provided by ol_query.

bad_query_string: More then 100 characters were placed in the query buffer.

bad_result_handle: Invalid result handle.

result_overflow: The query results overflowed the 30,000 character limit. The handle is still valid and contains only complete lines.

If an unexpected error is encountered then the value returned from ol_search will be a Macintosh OS error.

ol_term:

```
void ol_term (Handle ol_state);
PROCEDURE ol_term(ol_state: HANDLE); C;
```

Call ol_term when your application terminates or when you are finished using the On Location search engine. ol_state disposes of the state data initialized by ol_init.

NOTE: The handle is disposed by ol_term and any further use of it will be an error.

Build instructions:

The file ol_search_glue.o contains stub routines that will load the resource found in the file ol_search_code. In order to build an executable using this package, link with the file ol_search_glue.o (or import it into the Think environment) and use “rez” (or resedit or Think’s built in resource inclusion mechanism) to include the resource from ol_search_code (TYPE=ONLC, ID=0) into the executable. We have had success using the engine from Apple’s Allegro Common Lisp environment, call us if you would like more information.

HyperCard Users:

In order to use the search engine from HyperCard you must first copy the XFCN’s from the file :Examples:ol_search_XFCN into the stack that you wish to use it from. Please see the example stack OLSearch for details on how

to access the XFCN’s. If further information is needed please call or post messages to our bulletin boards.

General Information and limitations:

In this release simultaneous queries on the same index by the search engine and the On Location product are not supported.

If background indexing is enabled on the host machine it will be disabled when on_init is called and re-enabled with a call to on_term.

Remember that the search engine needs memory to run (so grow the memory partition of the host accordingly).

Contact:

Kim Agricola
On Technology, Inc
155 Second Street
Cambridge, MA 02141
(617) 876-0900
America Online: ON Tech