# ECM Library Content Management System

Developer: Dale Miller

Reviewer: Liz Garnand

Last Review Date:

# ECM Library

**Purpose**

ECM Library archives, manages, and provides easy access to your company’s intellectual and critical asset: all the electronic data. It creates one easy repository for your content, while maintaining control on who has access to what information. Using features that were formerly only affordable by Fortune 100 companies, ECM Library is affordable and intuitive for even the smallest business. Its architecture is based on decades of software design experience developing and deploying content enterprise solutions for the most complex and demanding organizations. This design team experience is combined with today’s leading-edge technologies and methodologies.

Search and retrievals are lightening fast, thorough and intelligent. The power and complexity are under-the-hood: users will experience an easy and intuitive Windows-based graphic user interface that integrates into Windows Office and Windows Explorer using familiar drag-and-drops and mouse-clicks.

* 1. What it provides
  2. What it saves
  3. E-Discovery
  4. Graphic/Picture Recognition
  5. Multimedia
     1. Music
     2. Voice Recordings (analog/digital)
     3. Video

# Licensing

Licenses are issued on a “per machine basis”. Any machine can be defined to the system up to the assigned number of licenses purchased. Then, an equal or less number of users can be assigned to the repository – up to one per machine.

Then the users can login and use any machine that is defined to the repository under their ID if it too is assigned to the repository. This allows for a user to go to a remote site and use the application provided he/she is defined to the repository and the machine he/she wishes to use is also defined to the repository. A machine is automatically entered into the repository when the application is installed and executed for the first time if the maximum number of seats (machines allowed) is not exceeded. If that number is exceeded, this machine cannot log into the system until a seat is freed up or more seats are purchased.

# Prerequisites

#### SQLDMO

Go to link <http://www.microsoft.com/downloads/details.aspx?FamilyID=d09c1d60-a13c-4479-9b91-9e8b9d835cdc&DisplayLang=en> and download if needed. You will have to have SQLDMO installed on all machines that run ECM Library.

The above will install Feature Pack for Microsoft SQL Server 2005 - November 2005

To register the SQL-DMO components on a client computer

From C:\Program Files\Microsoft SQL Server \90\Tools\Binn\Resources\<language> directory, execute: (or the directory in which the DLL resides as shown below)

%Program Files%\Microsoft SQL Server\80\Tools\Binn\REGSVR32 SQLDMO.DLL

#### SQL Server 2008

If you do not have SQL Server 2005 Enterprise or 2008 Enterprise, it will be necessary to download and install SQL Server 2008 Express with advanced Services. This is a free product from Microsoft and can be downloaded at <http://www.microsoft.com/express/sql/download/>.

#### Office 2003 or Office 2007

Either version of Microsift Office will work, but 2007 is preferred. One of them must be installed on the computer that will run ECM Library.

# Archiving

* 1. Emails
  2. Content
  3. Contacts

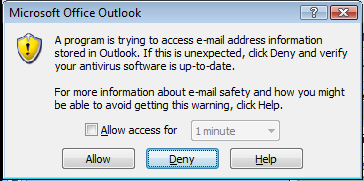
# Search

* 1. Methods
     1. Contains
     2. Free Text (Business Meaning, synonyms
        1. When FREETEXT is used, the search engine internally performs the following actions on the *freetext string*, assigns each term a weight, and then finds the matches.
           1. Separates the string into individual words based on word boundaries (word-breaking).
           2. Generates inflectional forms of the words (stemming).
           3. Identifies a list of expansions or replacements for the terms based on matches in the thesaurus.
     3. Weighted Results
        1. Can be used either in Contains or FREETEXT searches
        2. A minimum return weight can be specified
        3. Sorts the results in the most likely first and the least likely last
  2. Syntax
     1. The syntax of the search, or the way in which one enters words and phrases for which to search, is based on a common and accepted format. It will be best and most easily explained through example. For example, if a search were needed that looked for the single word “automobile” then all that would have to be entered into the search line would be the single word “automobile” and the search key pressed or the enter key pressed. Then the search engine would examine every document, email and piece of metadata for the exact word “automobile”. It goes without too much explanation that the single word search is simple and will return, at times, unmanageable amounts of content.
     2. To begin to limit the amount of returned data, let’s expand the search to include more key words to allow the search engine to reduce (be more selective) on what content is considered. So for this search, the terms “automobile accident” will be entered as separate words on the search command line. This instructs the search engine to look for content containing the word “accident” or “automobile”. This will most likely result in even more content being returned. There are times during searches when this is a desired effect, so remember, words separated with blanks are considered to be using the “OR” operation. Therefore, if I looked for the phrase “Albert Einstein”, I would bring back all the content containing the name “Albert’ or the name “Einstein”.
     3. It is a very easy process to change this search, either of the above searches to be much more specific. For example, entering automobile +accident in the search command line or Albert + Einstein would now instruct the search engine to bring back only content that contained both of the search words. I s indeed, very easy to begin to limit the amount of data being returned.
     4. A good example is when one of the larger insurance companies searched for robbins +fatal +car +accident +Kansas. This allowed the search engine to immediately reduce the set of returned content to just a few documents. The research analyst thought that this was too few understanding the magnitude of the case she was researching. She restructured the search criteria to read like this (and we will explain fully in a moment). The search she entered was robbins NEAR Collins +^fatal +^car +accident +Kansas +"Fort Riley". This returned less than 50 emails, documents, pictures, and filings out of some 1,000,000 documents. What this researcher had told the search engine to do was to find all content that contained the word “robbins” in close proximity to the word “Collins” (as it was Robbins vs. Collins in the filings), to search for words that had the meaning of fatal (such as death, casualty, fatality, etc.), include content that used the word auto, automobile, vehicle, etc., and contained the word accident within the document as well as the word Kansas (no consideration to capitalization) and contained the exact phrase “Fort Riley”. Once the search was initiated, the documents were returned for her to review in a couple of seconds. She reviewed these documents, selected the ones she determined to be pertinent and loaded them into a library she created for this case. She then added the other search team members to the library and passed the library link to the legal department.
     5. Wild card searches can be easily performed in case one is not sure of the full spelling of a word or wants to include all the words that may begin with, end with, or contain a phrase. For example if the spelling of Fort Robbins was not certain, the researcher could look for “Fort Robb\*” finding all occurances of any phrases that contained the starting letters “fort robb” and then had anything or nothing following them. The reverse is true also. In case you were unsure if it was Fort or Ft. a search cound be performed on +\*F\* +Robbins or “F\* Robbins” or “F\* Robb\*”. The power and flexibility of the ECM Library search engine is fully under the users’ control.
     6. The Last example deals with a phrase that you which to search considering its business meaning***. NOW THIS IS WHERE I HAVE TO THINK OF HOW BEST TO EXPLAIN THIS CONCEPT AS IT IS THE MOST ESOTERIC***.
  3. Email
  4. Content Search
* One or more specific words or phrases (*simple term*)
* A word or a phrase where the words begin with specified text (*prefix term*)
* Inflectional forms of a specific word (*generation term*)
* A word or phrase close to another word or phrase (*proximity term*)
* Synonymous forms of a specific word (*thesaurus*)
* Words or phrases using weighted values (*weighted term*)
  1. Global
  2. Screens
  3. Search History Table

# Unique Data Requirements

* 1. Library Names
  2. Group Names

# Components

Secure Access  


1. Purpose:
2. Components:
   1. Allow Access for
   2. Allow
   3. Deny
   4. Help

## The Control Screen

## 

* + 1. Select
    2. Tools
    3. Windows
    4. Help

## The Setup and Archive Management Screen

### Basic Description

#### Purpose:

1. Compoments:
   1. Email Archive Group:
      1. The displayed list
      2. Archive Emails in Folder
      3. Do not delete unread emails
      4. Remove after archive
      5. Archive items after
      6. Remove items after
      7. ALL Button
      8. Active Button
      9. Archive Button
      10. Use last folder date
   2. Polling Setup Group
      1. Run Archiver
      2. Every
      3. Load at startup
      4. Disable
      5. Save
   3. File Types Group
      1. Drop down list of file types
      2. The “\*” Button
      3. The “X” button
      4. Process Type
      5. As Type
      6. The List of assigned types
      7. Add Button
      8. Remove Button
   4. File Archive Group
      1. List of Active Directories
      2. Include all file types
      3. Available
      4. Include
      5. Exclude
      6. Select Dir
      7. Include Dir
      8. Remove Dir
      9. Save Change
      10. Refresh
      11. Include Sub-directories
      12. Make Public
      13. Disable Dir Archive
      14. Version Files
      15. Capture Metadata
      16. Include Button
      17. Exclude Button
      18. Remove under Include List
      19. Remove under exclude List
      20. Skip if Archive bit is on
      21. Archive Content Button
   5. Progress Bar
   6. Status Bar
   7. Once Now Button

## E-Mail Search Screen

## 

|  |  |  |
| --- | --- | --- |
| **WidgetText** | **WidgetName** | **HelpText** |
|  | btnSearchFrom | <help text here> |
|  | Button3 | <help text here> |
|  | Button4 | <help text here> |
|  | Button5 | <help text here> |
|  | Button6 | <help text here> |
|  | cbCCaddr | This will contain a drop down list of CC addresses |
|  | cbDateSelection | A drop down of availalbe CC`s in the repository. |
|  | cbFolderFilter | This will contain a drop down list of searchable email folders |
|  | cbFromAddr | This will contain a drop down list of searchable email from addresses |
|  | cbFromName | This will contain a drop down list of searchable email from names |
|  | cbLibrary | A list of available libraries. |
|  | cbSavedEmailSearches | This will contain a drop down list of saved email searches. |
|  | cbToAddr | This will contain a drop down list of searchable email to Addresses. |
|  | cbToName | This will contain a drop down list of searchable email to names. |
|  | dgAttachments | The list of associated attachments for the selected email. |
|  | dgSearch | The search return results. |
|  | SB\_Warning | Warning messges. |
|  | SB2 | Status bar |
|  | txtBody | This is the search text for the subject and the body of the email. |
|  | txtCCPhrase | This searches the CC`d list of people for the email and is wildcard able. |
|  | txtDays | <help text here> |
|  | txtFromAddr | This searches the from address of the email and is wildcard able. |
|  | txtFromName | This searches the from name of the email and is wildcard able. |
|  | PB | <help text here> |
|  | txtSearch | <help text here> |
|  | txtSubject | This searches the subject of the email and is wildcard able. |
|  | txtThesaurus | <help text here> |
|  | txtToAddr | This searches the to address of the email and is wildcard able. |
|  | txtToName | This searches the to name of the email and is wildcard able. |
| #Found | SB | Status Bar |
| 0 | nbrWeightMin | <help text here> |
| 1 | nbrSelectSearch | <help text here> |
| All Emails | rbAllEmails | Check to searh ALL Emails that are shared, public, or in a library you have access to. |
| AND | cbThesaurus | <help text here> |
| Append Library | btnLibrarySave | Append selected items to the selected Library |
| Business Meaning | ckBusiness | Check to search on the Business Meaning of the search criteria. |
| C | btnCopy | Press to copy thes earch line into the clipboard. |
| C | Button9 | <help text here> |
| Check Count | ckCount | Check Count to have the number of rows returned calculated. |
| Creation | ckCreationDate | <help text here> |
| days. | Label16 | <help text here> |
| Delete Search | btnDelSearch | Delete the selected Search |
| Folder Filter: | Label15 | <help text here> |
| Gen Sql | btnGenSql | Press to Generate the Sql stmt and put it into the clipboard. |
| Limit to created or sent within the last | ckDays | <help text here> |
| Limit to current list | ckLimitToExisting | Limit to current list of found emails. |
| My Emails | rbMyEmails | Check to searh only My Emails |
| or where contains the phrase: | Label13 | <help text here> |
| or where contains the phrase: | Label3 | <help text here> |
| or where contains the phrase: | Label4 | <help text here> |
| or where contains the phrase: | Label7 | <help text here> |
| or where contains the phrase: | Label8 | <help text here> |
| P | Button8 | <help text here> |
| P | btnPaste | Press to paste the clipboard into the search line. |
| Recall Search | btnRecall | Recall selected Search items |
| Received | ckReceivedTime | <help text here> |
| Received | rbRx | Check to search Received items |
| Restore | btnRestore | Restore the selected items |
| Return only those that have attachments | ckOnlyWithAttach | Return only those that have attachments |
| S | btnSpellCk | Press to spell check the search line. |
| S | Button7 | <help text here> |
| Save Search | btnSave | Save the selected Search |
| Search | btnSearch | Excute the Search |
| Search Subject for: | Label5 | <help text here> |
| Search by CC/BCC email addr: | Label14 | <help text here> |
| Search by FROM email addr: | Label1 | This is the label for the Search by FROM email addr. |
| Search by FROM Name: | Label2 | <help text here> |
| Search by MAIL Date: | Label6 | <help text here> |
| Search by TO email addr: | Label10 | <help text here> |
| Search by TO Name: | Label9 | <help text here> |
| Search Email Attachments | ckIncludeAttachments | Check to Search Email Attachments as part of the search. |
| Sent | ckSentDate | <help text here> |
| Sent | rbSent | Check to search Sent items |
| show items with weight greater >= | lblWeightLimit | <help text here> |
| Show Search Details | ckShowDetails | Show Search Details window |
| Show Weights | ckWeighted | Show Weights of the search and order by the weighted return value. |
| Test Display | Button1 | Test Display: this will go away |
| The Qry | txtQry | The generated Qry |
| Thesaurus: | Label11 | <help text here> |
| Tuesday | dtMailDateEnd | This is the start date. |
| Tuesday | dtMailDateStart | This is the end date. |
| Where Contains | Label12 | <help text here> |
| Working Dir. | Button2 | Open the Working Dir. |

## Content Search Screen

Search Content For

Limit Search to Master Docs

## 

### The Right Mouse Menu

* + - 1. Restore Selected Documents
      2. Add selected documents to library
      3. Group Management Screen
      4. Library/User management
      5. Content Management Settings

#### Callable from All Search Screens

* + - 1. User Notifications: User Reassignment

## Search All of the Repository

Purpose:

Screen Components:

* MIDI Parent:
* Working Directory Content
* Working Directory Emails
* Grid Display
* Search Bar

|  |  |  |
| --- | --- | --- |
| WidgetText | WidgetName | HelpText |
|  | cbLibrary | A drop down list of available Libraries |
|  | dgGlobalSearch | Search results |
|  | nbrWeightMin | Enter the minimum weight that will be returned. |
|  | PB | Progress bar |
|  | TempGrid | <help text here> |
|  | txtContents | <help text here> |
|  | txtSearch | Enter search parameters here |
|  | txtSelDir | Displays the selected directory |
|  | txtThesaurus | <help text here> |
|  | SB | Status bar |
| &Restore | btnRestore | Press toRestore selected |
| &Search | btnSearch | Press to initiate Search |
| 20 | nbrSelectSearch | The nuber of the previous seraches you have performed, change to select the desired previous search. |
| Add to Library | btnLibrary | Press to add the selected to the Library |
| All Content | rbAll | Pres to select both email and document Content |
| AND | cbThesaurus | <help text here> |
| Business Meaning | ckBusiness | Check to use Business Meaning as the level of search |
| C | btnCopy | Copy search line into clipboard. |
| C | Button3 | <help text here> |
| Documents | rbDocs | Pres to select document Content |
| Emails | rbEmails | Pres to select email Content |
| Gen Sql | btnGenSql | Press to generate the sql statement that will be execturd and place it into the clipboards. |
| Get counts only | ckCountOnly | Check to Get counts only |
| Label1 | Label1 | <help text here> |
| Library Management | btnLibMgt | Press to open Library Management |
| Limit to current list | ckLimitToExisting | Check to Limit to current list displayed items |
| Only show items with a weight greater than or equal to: | lblWeightLimit | Only show items with a weight greater than or equal to: |
| Open Restored Content Dir | btnWorkingDirDocs | Press to Open Restored Content Dir |
| Open Restored Emails Dir | btnWorkingDir | Press to Open Restored Emails Dir |
| Overwrite | ckOverWrite | Check to overwrite existing files during restore. |
| P | btnPaste | Copy clipboard into search line. |
| P | Button2 | <help text here> |
| Quick search does not search email attachments and only on content within an email or file. | Label2 | <help text here> |
| Refresh | btnRefreshLibrary | Press to Refresh the libraries |
| S | Button1 | <help text here> |
| S | btnSpellCk | Spell check search line. |
| Search For: | Label3 | <help text here> |
| Select Directory | selDir | Press to Select Directory for restore |
| Show Details | ckShowDetails | Press to display the Show Details screen |
| Show Weights | ckWeighted | Press to calcualte and Show Weights |
| Thesaurus: | Label4 | <help text here> |
| To default directory | rbToDefaultDir | Restore to default directory |
| To original directory | rbToOriginalDir | Restore to original directory of the doument |
| To selected directory | rbToSelDir | Restore to a selected directory |

## Library Creation and Management Screen

## User Groups

#### How groups work:

## 

#### User Notifications:

* + - * 1. Add to a group
        2. Removed from a group

## Library Items Management Screen

#### User Notifications:

* + - 1. Added to a library
      2. Removed from a library

## User Startup Parameters

## Working Folders

* + 1. Purpose
    2. Setup
    3. Purging

## File Types

* + 1. Purpose
    2. Default Types
    3. New Types

## The Users Setup

## Contact Management Screens Navigation

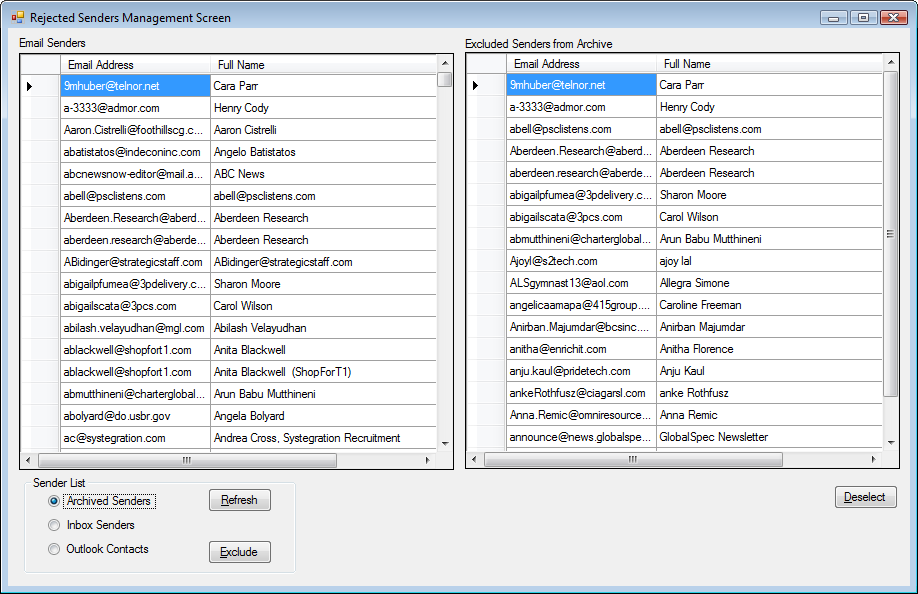
### Archive Contacts

### 

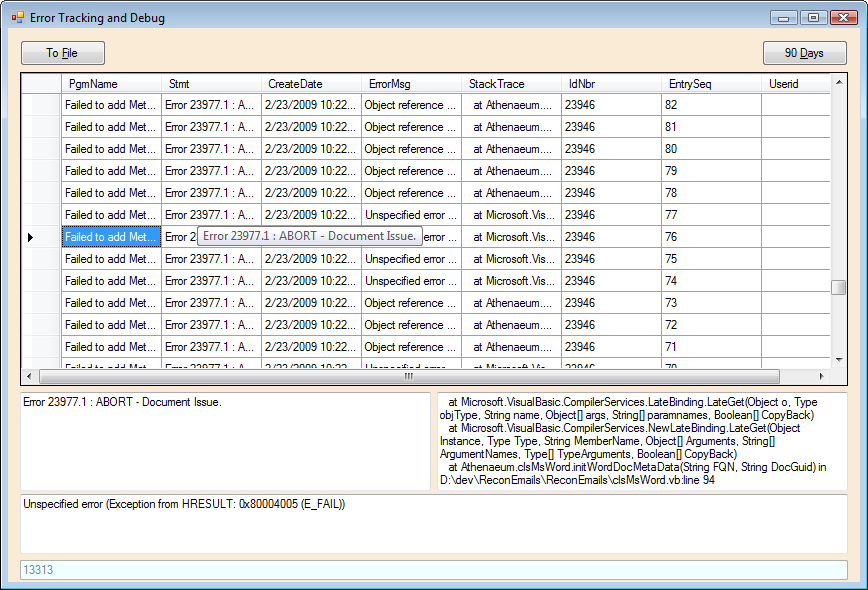
* + 1. Archive Contacts:  
       There is no screen for this action. It initiates a process.

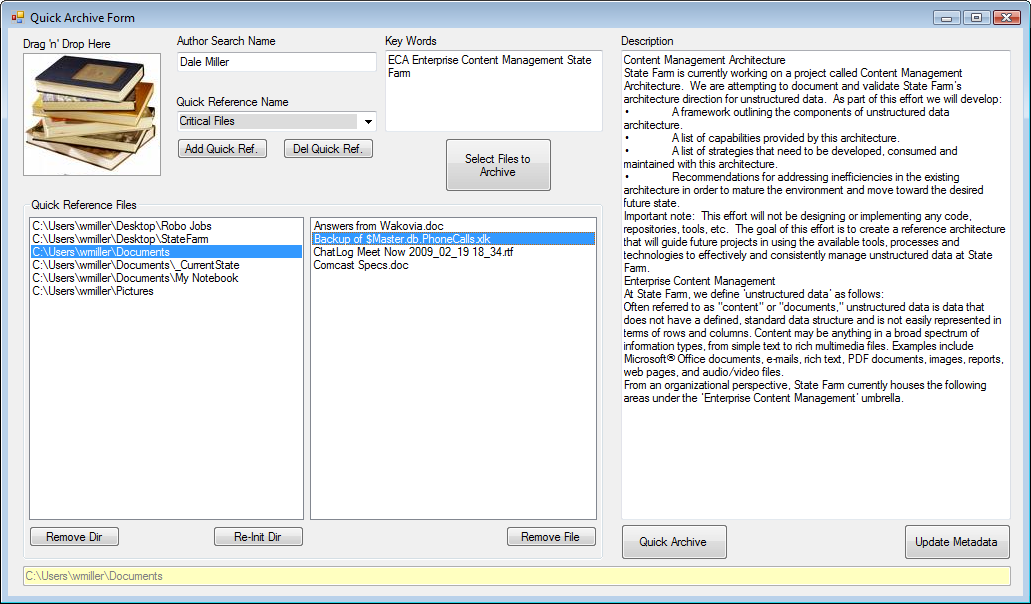
### Contact Management Screen

* Restore Selected
  + Overwrite if exists
  + Skip if exists
  + Add if missing
* Delete Selected
* Archive Append
* Replace All

Reject Senders Screen  


Email Senders

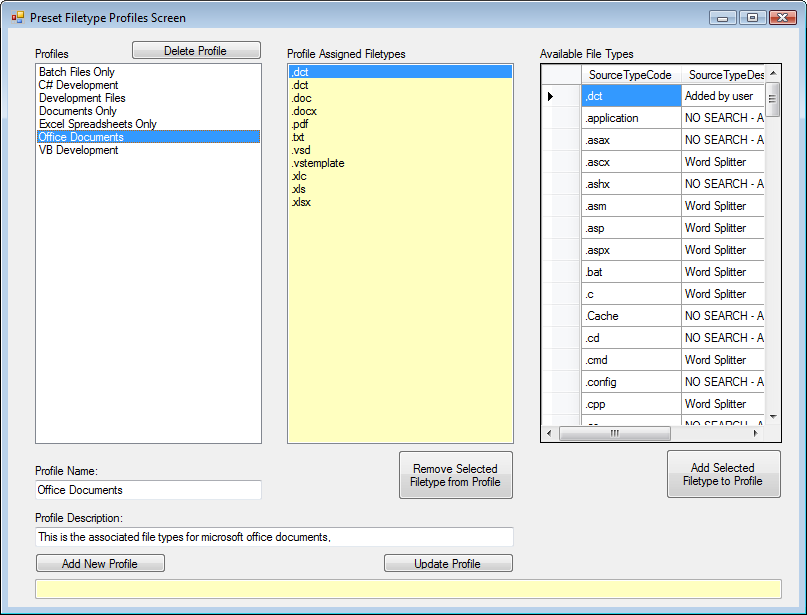
Error Tracking and Reporting  


Quick Archive  


## Login as a Different User

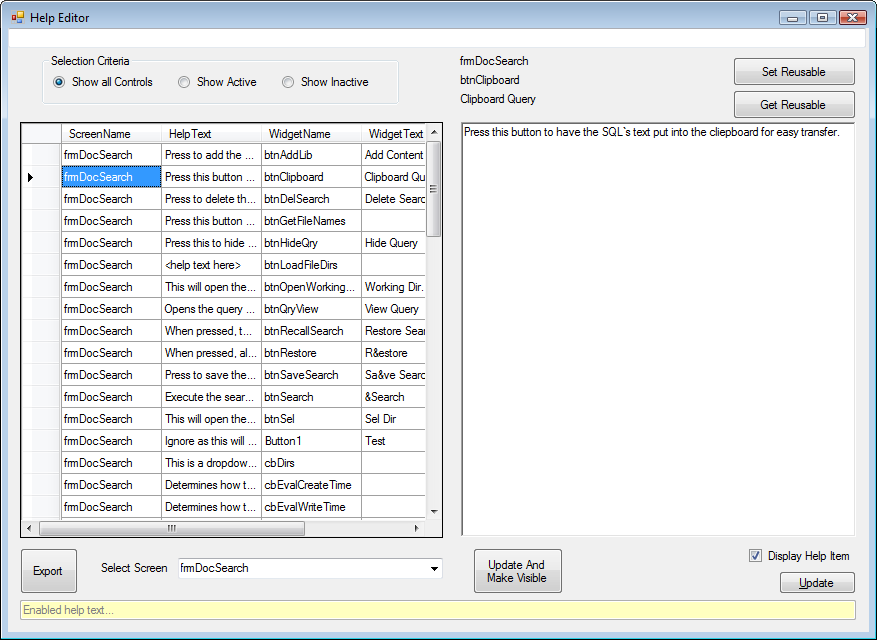
* + 1. Purpose
    2. Limitations
    3. Login ID
    4. Components
       1. Password
       2. OK
       3. Cancel

## Profile Maintenance

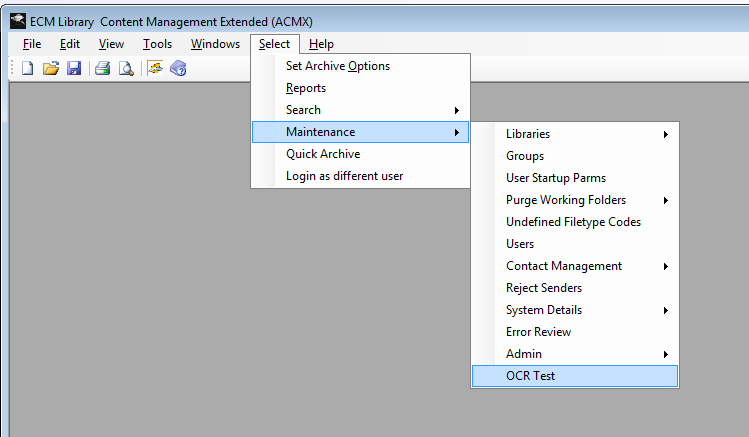


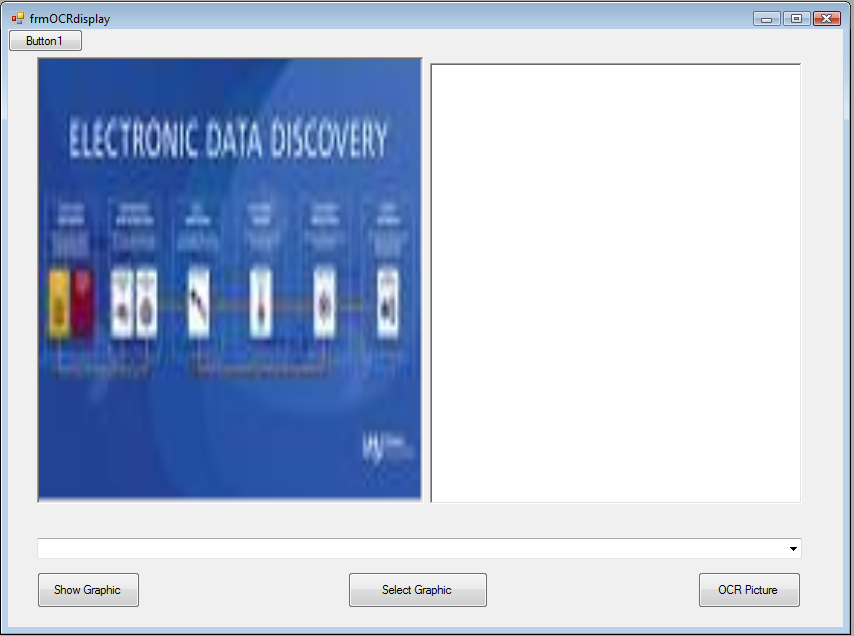
* + - * Purpose
      * Limitations
      * Components
        + Profiles
        + Profile Assigned Types
        + Available Filet ype
        + Profile Name
        + Profile Description
        + Add New Profile
        + Update Profile
        + Remove Selected Filetype from Profile
        + Add Selected Filetype to Profile
        + Status Bar

# Help Editor

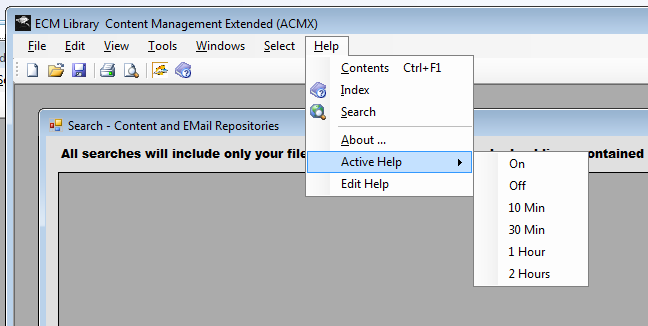


# OCR Test

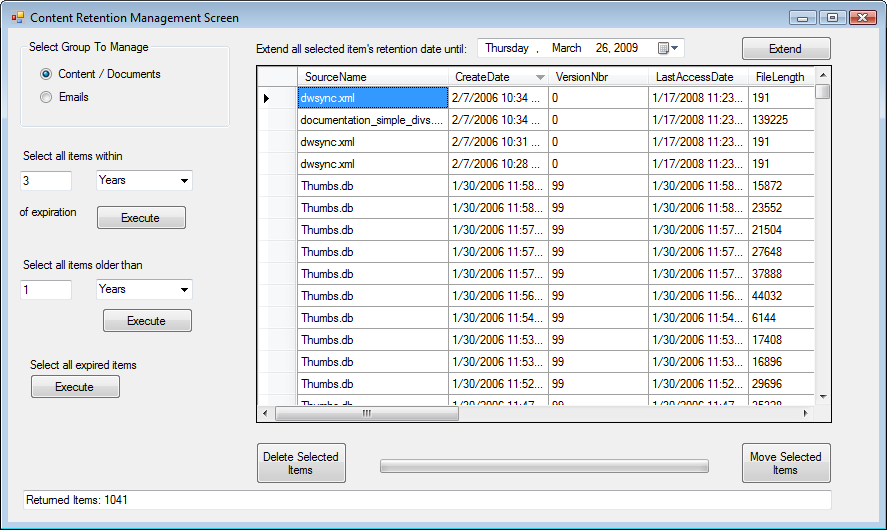




# Active Help



## Retention Management



# Setup

* 1. The database
     1. Initial User
     2. Initial Settings

## APP.Config File

## Web Install

* + 1. How to setup the Web Install
    2. How to execute the Web Install
    3. What the install adds to your system

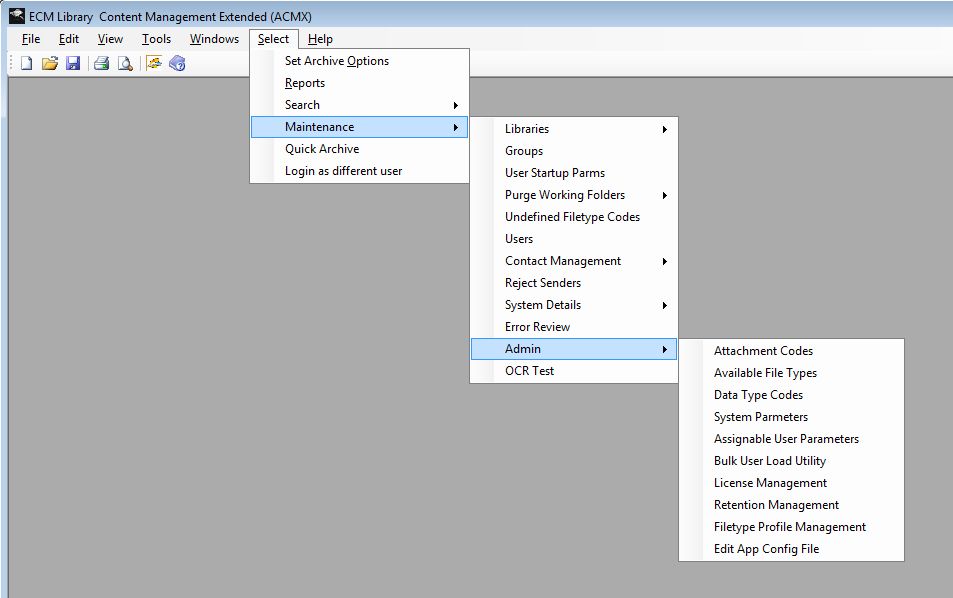
## Minimum System Requirements

* + 1. Operating Systems
    2. Processor
    3. Memory
    4. Databases
       1. SQL Server 2005
       2. SQL Server 2008
       3. SQL Server 2005 Desktop
       4. SQL Server 2008 Desktop

## Product Utilization

* + 1. Enterprise Wide
    2. Single User
       1. Synchronization
       2. Limited Search When not Connected to Net

# Admin Functions



* Attachment Codes
* Available File Types
* Data Type Codes
* System Parameters
* Assignable User Parameters
* Bulk User Load Utility
* License Management
* Retention Management
* Filetype Profile Management
* Edit App Configuration File

# Reference Software

* SQL Server 2005 or greater
* SQL Express Edition
* SQL Server Management Studio
* SQL Server Management Studio Express
* The ECM Library repository

# Image XMP Tagging

XMP information is rapidly becoming the standard, working in conjunction with XML, for embedding metadata into images. ECM Library provides users with the capability to embedded needed metadata within their digital pictures. ECM Library will also scan, extract and archive for search any embedded XMP data within digital pictures.

The Adobe Extensible Metadata Platform (XMP) is a standard for processing and storing standardized and proprietary metadata, created by Adobe Systems Inc.

XMP standardizes the definition, creation, and processing of extensible metadata. Serialized XMP can be embedded into a significant number of popular file formats, without breaking their readability by non-XMP-aware applications. Embedding metadata ("the truth is in the file") avoids many problems that occur when metadata is stored separately. XMP is used in PDF, photography and photo editing applications.

### XMP Data Model

XMP defines a metadata model that can be used with any defined set of metadata items. XMP also defines particular schemas for basic properties useful for recording the history of a resource as it passes through multiple processing steps, from being photographed, scanned, or authored as text, through photo editing steps (such as cropping or color adjustment), to assembly into a final image. XMP allows each software program or device along the way to add its own information to a digital resource, which can then be retained in the final digital file.

XMP is most commonly serialized and stored using a subset of the W3C Resource Description Framework (RDF), which is in turn expressed in XML.

# Appendix I: The Search Engine Syntax

### CONTAINS

      ( { column\_name | ( column\_list ) | \* }

          , '< contains\_search\_condition >'

   [ , LANGUAGE language\_term ]

      )

< contains\_search\_condition > ::=

    { < simple\_term >

    | < prefix\_term >

    | < generation\_term >

    | < proximity\_term >

    | < weighted\_term >

    }

    | { ( < contains\_search\_condition > )

    [ { < AND > | < AND NOT > | < OR > } ]

< contains\_search\_condition > [ ...n ]

    }

< simple\_term > ::=

          word | " phrase "

< prefix term > ::=

     { "word \* " | "phrase \*" }

< generation\_term > ::=

     FORMSOF ( { INFLECTIONAL | THESAURUS } , < simple\_term > [ ,...n ] )

< proximity\_term > ::=

     { < simple\_term > | < prefix\_term > }

     { { NEAR | ~ }

     { < simple\_term > | < prefix\_term > }

     } [ ...n ]

< weighted\_term > ::=

     ISABOUT

        ( { {

  < simple\_term >

  | < prefix\_term >

  | < generation\_term >

  | < proximity\_term >

  }

   [ WEIGHT ( weight\_value ) ]

   } [ ,...n ]

        )

< AND > ::=

     { AND | & }

< AND NOT > ::=

     { AND NOT | & !}

< OR > ::=

     { OR | | }

### http://i.msdn.microsoft.com/Global/Images/clear.gif Arguments

*column\_name*

Is the name of a full-text indexed column of the table specified in the FROM clause. The columns can be of type **char**, **varchar**, **nchar**, **nvarchar**, **text**, **ntext**, **image**, **xml**, **varbinary**, or **varbinary(max)**.

*column\_list*

Specifies two or more columns, separated by commas. *column\_list* must be enclosed in parentheses. Unless *language\_term* is specified, the language of all columns of *column\_list* must be the same.

\*

Specifies that the query will search all full-text indexed columns in the table specified in the FROM clause for the given search condition. The columns in the CONTAINS clause must come from a single table that has a full-text index. Unless *language\_term* is specified, the language of all columns of the table must be the same.

LANGUAGE *language\_term*

Is the language to use for word breaking, stemming, thesaurus expansions and replacements, and noise-word (or) removal as part of the query. This parameter is optional.

If documents of different languages are stored together as binary large objects (BLOBs) in a single column, the locale identifier (LCID) of a given document determines what language to use to index its content. When querying such a column, specifying LANGUAGE *language\_term* can increase the probability of a good match.

*language\_term* can be specified as a string, integer, or hexadecimal value corresponding to the LCID of a language. If *language\_term* is specified, the language it represents will be applied to all elements of the search condition. If no value is specified, the column full-text language is used.

When specified as a string, *language\_term* corresponds to the **alias** column value in he compatibility view. The string must be enclosed in single quotation marks, as in '*language\_term*'. When specified as an integer, *language\_term* is the actual LCID that identifies the language. When specified as a hexadecimal value, *language\_term* is 0x followed by the hexadecimal value of the LCID. The hexadecimal value must not exceed eight digits, including leading zeros.

If the value is in double-byte character set (DBCS) format, SQL Server will convert it to Unicode.

If the language specified is not valid or there are no resources installed that correspond to that language, SQL Server returns an error. To use the neutral language resources, specify 0x0 as *language\_term*.

<*contains\_search\_condition*>

Specifies the text to search for in *column\_name* and the conditions for a match.

*<contains\_search\_condition>* is **nvarchar**. An implicit conversion occurs when another character data type is used as input. In the following example, the @SearchWord variable, which is defined as varchar(30), causes an implicit conversion in the CONTAINS predicate.

USE AdventureWorks;

GO

DECLARE @SearchWord varchar(30)

SET @SearchWord ='performance'

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, @SearchWord);

Because "parameter sniffing" does not work across conversion, use **nvarchar** for better performance. In the example, declare @SearchWord as nvarchar(30).

USE AdventureWorks;

GO

DECLARE @SearchWord nvarchar(30)

SET @SearchWord = N'performance'

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, @SearchWord);

You can also use the OPTIMIZE FOR query hint for cases in which a non optimal plan is generated.

*word*

Is a string of characters without spaces or punctuation.

*phrase*

Is one or more words with spaces between each word.

|  |
| --- |
| **Note:** |
| Some languages, such as those written in some parts of Asia, can have phrases that consist of one or more words without spaces between them. |

<simple\_term>

Specifies a match for an exact word or a phrase. Examples of valid simple terms are "blue berry", blueberry, and "Microsoft SQL Server". Phrases should be enclosed in double quotation marks (""). Words in a phrase must appear in the same order as specified in *<contains\_search\_condition>* as they appear in the database column. The search for characters in the word or phrase is not case-sensitive. Noise words (such as a, and, or the) in full-text indexed columns are not stored in the full-text index. If a noise word is used in a single word search, SQL Server returns an error message indicating that the query contains only noise words. SQL Server includes a standard list of noise words in the directory \Mssql\Binn\FTERef of each instance of SQL Server.

Punctuation is ignored. Therefore, CONTAINS(testing, "computer failure") matches a row with the value, "Where is my computer? Failure to find it would be expensive." For more information on word-breaker behavior, see.

<prefix\_term>

Specifies a match of words or phrases beginning with the specified text. Enclose a prefix term in double quotation marks ("") and add an asterisk (\*) before the ending quotation mark, so that all text starting with the simple term specified before the asterisk is matched. The clause should be specified this way: CONTAINS (column, '"text\*"'). The asterisk matches zero, one, or more characters (of the root word or words in the word or phrase). If the text and asterisk are not delimited by double quotation marks, so the predicate reads CONTAINS (column, 'text\*'), full-text search considers the asterisk as a character and searches for exact matches to text\*. The full-text engine will not find words with the asterisk (\*) character because word breakers typically ignore such characters.

When *<prefix\_term>* is a phrase, each word contained in the phrase is considered to be a separate prefix. Therefore, a query specifying a prefix term of "local wine\*" matches any rows with the text of "local winery", "locally wined and dined", and so on.

<generation\_term>

Specifies a match of words when the included simple terms include variants of the original word for which to search.

INFLECTIONAL

Specifies that the language-dependent stemmer is to be used on the specified simple term. Stemmer behavior is defined based on stemming rules of each specific language. The neutral language does not have an associated stemmer. The column language of the columns being queried is used to refer to the desired stemmer. If *language\_term* is specified, the stemmer corresponding to that language is used.

A given *<simple\_term>* within a *<generation\_term>* will not match both nouns and verbs.

THESAURUS

Specifies that the thesaurus corresponding to the column full-text language, or the language specified in the query is used. The longest pattern or patterns from the *<simple\_term>* are matched against the thesaurus and additional terms are generated to expand or replace the original pattern. If a match is not found for all or part of the *<simple\_term>*, the non-matching portion is treated as a *simple\_term*. For more information on the full-text search thesaurus, see.

<proximity\_term>

Specifies a match of words or phrases that must be in the document that is being searched. Like the AND operator, *<proximity\_term>* requires both the search terms to exist in the document being searched.

NEAR | ~

Indicates that the word or phrase on each side of the NEAR or ~ operator must occur in a document for a match to be returned. Several proximity terms can be chained, as in **a NEAR b NEAR c** or a ~ b ~ c. Chained proximity terms must all be in the document for a match to be returned.

When used in the CONTAINSTABLE function, the proximity of the search terms affects the ranking of each document. The nearer the matched search terms are in a document, the higher the ranking of the document. If matched search terms are >50 terms apart, the rank returned on the document is 0.

For example, CONTAINS (*column\_name*, 'fox NEAR chicken') and CONTAINSTABLE (*table\_name*, *column\_name*, 'fox ~ chicken') would both return any documents in the specified column that contain both "fox" and "chicken". In addition, CONTAINSTABLE returns a rank for each document based on the proximity of "fox" and "chicken". For example, if a document contains the sentence, "The fox ate the chicken," its ranking would be high.

NEAR indicates the logical distance between terms, rather than the absolute distance between them. For example, terms within different phrases or sentences within a paragraph are treated as farther apart than terms in the same phrase or sentence, regardless of their actual proximity, on the assumption that they are less related. Likewise, terms in different paragraphs are treated as being even farther apart.

<weighted\_term>

Specifies that the matching rows (returned by the query) match a list of words and phrases, each optionally given a weighting value.

ISABOUT

Specifies the *<weighted\_term>* keyword.

WEIGHT(*weight\_value*)

Specifies a weight value, which is a number from 0.0 through 1.0. Each component in *<weighted\_term>* may include a *weight\_value*. *weight\_value* is a way to change how various portions of a query affect the rank value assigned to each row matching the query. WEIGHT does not affect the results of CONTAINS queries, but WEIGHT impacts rank in queries.

|  |
| --- |
| **Note:** |
| The decimal separator is always a period, regardless of the operating system locale. |

{ AND | & } | { AND NOT | &! } | { OR | | }

Specifies a logical operation between two contains search conditions.

AND | &

Indicates that the two contains search conditions must be met for a match. The ampersand symbol (&) may be used instead of the AND keyword to represent the AND operator.

AND NOT | &!

Indicates that the second search condition must not be present for a match. The ampersand followed by the exclamation mark symbol (&!) may be used instead of the AND NOT keyword to represent the AND NOT operator.

OR | |

Indicates that either of the two contains search conditions must be met for a match. The bar symbol (|) may be used instead of the OR keyword to represent the OR operator.

When *<contains\_search\_condition>* contains parenthesized groups, these parenthesized groups are evaluated first. After evaluating parenthesized groups, these rules apply when using these logical operators with contains search conditions:

* NOT is applied before AND.
* NOT can only occur after AND, as in AND NOT. The OR NOT operator is not allowed. NOT cannot be specified before the first term. For example, CONTAINS (mycolumn, 'NOT "phrase\_to\_search\_for" ' ) is not valid.
* AND is applied before OR.
* Boolean operators of the same type (AND, OR) are associative and can therefore be applied in any order.

*n*

Is a placeholder indicating that multiple CONTAINS search conditions and terms within them can be specified.

http://i.msdn.microsoft.com/Global/Images/clear.gif Remarks

Full-text predicates and functions work on a single table, which is implied in the FROM predicate. To search on multiple tables, use a joined table in your FROM clause to search on a result set that is the product of two or more tables.

CONTAINS is not recognized as a keyword if the compatibility level is less than 70. For more information, see.

Full-text predicates are not allowed in the when the database compatibility level is set to 100.

## Comparison of LIKE to Full-Text Search

In contrast to full-text search, the Transact-SQL predicate works on character patterns only. Also, you cannot use the LIKE predicate to query formatted binary data. Furthermore, a LIKE query against a large amount of unstructured text data is much slower than an equivalent full-text query against the same data. A LIKE query against millions of rows of text data can take minutes to return; whereas a full-text query can take only seconds or less against the same data, depending on the number of rows that are returned.

http://i.msdn.microsoft.com/Global/Images/clear.gif Examples

### A. Using CONTAINS with <simple\_term>

The following example finds all products with a price of $80.99 that contain the word "Mountain".

USE AdventureWorks;

GO

SELECT Name, ListPrice

FROM Production.Product

WHERE ListPrice = 80.99

AND CONTAINS(Name, 'Mountain');

GO

### B. Using CONTAINS and phrase in <simple\_term>

The following example returns all products that contain either the phrase "Mountain" or "Road".

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE CONTAINS(Name, ' "Mountain" OR "Road" ')

GO

### C. Using CONTAINS with <prefix\_term>

The following example returns all product names with at least one word starting with the prefix chain in the Name column.

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE CONTAINS(Name, ' "Chain\*" ');

GO

### D. Using CONTAINS and OR with <prefix\_term>

The following example returns all category descriptions containing strings with prefixes of either "chain" or "full".

USE AdventureWorks;

GO

SELECT Name

FROM Production.Product

WHERE CONTAINS(Name, '"chain\*" OR "full\*"');

GO

### E. Using CONTAINS with <proximity\_term>

The following example returns all product names that have the word bike near the word performance.

USE AdventureWorks;

GO

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, 'bike NEAR performance');

GO

### F. Using CONTAINS with <generation\_term>

The following example searches for all products with words of the form ride: riding, ridden, and so on.

USE AdventureWorks;

GO

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, ' FORMSOF (INFLECTIONAL, ride) ');

GO

### G. Using CONTAINS with <weighted\_term>

The following example searches for all product names containing the words performance, comfortable, or smooth, and different weightings are given to each word.

USE AdventureWorks;

GO

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, 'ISABOUT (performance weight (.8),

comfortable weight (.4), smooth weight (.2) )' );

GO

### H. Using CONTAINS with variables

The following example uses a variable instead of a specific search term.

USE AdventureWorks;

GO

DECLARE @SearchWord nvarchar(30)

SET @SearchWord = N'Performance'

SELECT Description

FROM Production.ProductDescription

WHERE CONTAINS(Description, @SearchWord);

GO

### I. Using CONTAINS with A Logical Operator (AND)

The following example uses the **ProductDescription** table of the **AdventureWorks** database. The query uses the CONTAINS predicate to search for descriptions in which the description ID is not equal to 5 and the description contains both the word "Aluminum" and the word "spindle." The search condition uses the AND Boolean operator.

USE AdventureWorks;

GO

SELECT Description

FROM Production.ProductDescription

WHERE ProductDescriptionID <> 5 AND

CONTAINS(Description, ' Aluminum AND spindle');

GO

### J. Using CONTAINS to Verify a Row Insertion

The following example uses CONTAINS within a SELECT subquery. Using the **AdventureWorks** database, the query obtains the comment value of all the comments in the **ProductReview** table for a particular cycle. The search condition uses the AND Boolean operator.

USE AdventureWorks;

GO

INSERT INTO Production.ProductReview

(ProductID, ReviewerName, EmailAddress, Rating, Comments)

VALUES

(780, 'John Smith', 'john@fourthcoffee.com', 5,

'The Mountain-200 Silver from AdventureWorks Cycles meets and exceeds expectations. I enjoyed the smooth ride down the roads of Redmond')

-- Given the full-text catalog for these tables is Adv\_ft\_ctlg,

-- with change\_tracking on so that the full-text indexes are updated automatically.

WAITFOR DELAY '00:00:30'

-- Wait 30 seconds to make sure that the full-text index gets updated.

SELECT r.Comments, p.Name

FROM Production.ProductReview r

JOIN Production.Product p

ON

 r.ProductID = p.ProductID

AND r.ProductID = (SELECT ProductID

                  FROM Production.ProductReview

                  WHERE CONTAINS (Comments,

                                 ' AdventureWorks AND

                                   Redmond AND

                                   "Mountain-200 Silver" '))

GO

## A. Using FREETEXT to search for words containing specified character values

The FREETEXT command is another alternative to access the data indexed by Full Text Search. In general the FREETEXT command provides the ability to search for a matched term based on the meaning of the terms as opposed to the exact character string. At a high level, this command finds matches based on separating the string into individual words, determining inflectional versions of the word and using a thesaurus to expand or replace the term to improve the search.

Now let's compare the FREETEXT functionality with the CONTAINS command. The CONTAINS command uses exact match or fuzzy logic to perform the matches with a single word or a phrase. In addition, the words near another word can be found as well as performing a weighted match of multiple words where each word has a weight as compared to the others that are searched. Check out CONTAINS (Transact-SQL) for a explanation on the CONTAINS command.

Depending on the search you are performing dictates which Full Text Search command you should use. Keep in mind that FREETEXT and CONTAINS are only two of the four commands available. The other two commands are CONTAINSTABLE and FREETEXTTABLE. The comparison between the four commands will be saved for a future tip since it is fairly involved explanation that should include examples.

***Until that point in time, here is one data point to consider: according to SQL Server 2005 Books Online FREETEXT (Transact-SQL) "Full-text queries using FREETEXT are less precise than those full-text queries using CONTAINS.*** The SQL Server full-text search engine identifies important words and phrases. No special meaning is given to any of the reserved keywords or wildcard characters that typically have meaning when specified in the <contains\_search\_condition> parameter of the CONTAINS predicate." Based on my testing, when the basic terms are queried with either command similar results are returned, so the precise factor for simple queries seems less of an issue. For complex searches the CONTAINS command wins hands down with the ability to use wild cards, NEAR statements, etc. As such, if the flexibility of the search is built into the front end application then the highest level of flexibility on the back end, between the FREETEXT and CONTAINS commands, tips the scales toward the CONTAINS command.

The following example searches for all documents containing the words ***related to*** vital, safety, components.

USE AdventureWorks;

GO

SELECT Title

FROM Production.Document

WHERE FREETEXT (Document, 'vital safety components' );

GO

# Appendix II: DBA Stored Procedures

There are stored procedures in each database that can assist you in gathering information you need to document or work with full-text catalogs already present in your environment. As with other stored procedures that provide information on objects in SQL Server, these stored procedures start with "sp\_help". The following table shows those stored procedures, their purpose, and information provided by the stored procedure:

|  |  |
| --- | --- |
| sp\_help\_fulltext\_catalogs Must be run in the database in which it resides | |
| USE adventureworks2000 GO  exec sp\_help\_fulltext\_catalogs @fulltext\_catalog\_name = 'ctgDocumentSummary' | |
| ftcatid | The ID of the full-text catalog |
| Name | The name of the full-text catalog |
| Path | The physical location of the Gatherer Project folder |
| Status | The current status of the catalog:  0 = Idle  1 = Full population in progress  2 = Paused  3 = Throttled  4 = Recovering  5 = Shutdown  6 = Incremental population in progress  7 = Building index  8 = Disk is full. Paused  9 = Change tracking |
| Number\_FullText\_Tables | The number of tables associated with the catalog |
| sp\_help\_fulltext\_catalogs\_cursor Must be run in the database in which it resides | |
| USE adventureworks2000  GO  DECLARE @mycursor CURSOR  EXEC sp\_help\_fulltext\_catalogs\_cursor @mycursor OUTPUT, 'ctgDocumentSummary'  FETCH NEXT FROM @mycursor  WHILE (@@FETCH\_STATUS <> -1)  BEGIN  FETCH NEXT FROM @mycursor  END  CLOSE @mycursor  DEALLOCATE @mycursor  GO | |
| ftcatid | The ID of the full-text catalog |
| Name | The name of the full-text catalog |
| Path | The physical location of the Gatherer Project folder |
| Status | The current status of the catalog:  0 = Idle  1 = Full population in progress  2 = Paused  3 = Throttled  4 = Recovering  5 = Shutdown  6 = Incremental population in progress  7 = Building index  8 = Disk is full. Paused  9 = Change tracking |
| Number\_FullText\_Tables | The number of tables associated with the catalog |
| sp\_help\_fulltext\_columns Must be run in the database in which it resides | |
| USE adventureworks2000 GO  exec sp\_help\_fulltext\_columns @table\_name = 'Document' (optional: @column\_name = 'column\_name')  Leaving @column\_name NULL returns all columns and associated information | |
| TABLE\_OWNER | The owner of the table |
| TABLE\_ID | ID of the table |
| TABLE\_NAME | The name of the table |
| FULLTEXT\_COLID | The ID of the column (from the table itself) |
| FULLTEXT\_COLUMN\_NAME | Column of the table that specifies the document type (applicable only when data type is *image*) |
| FULLTEXT\_BLOBTP\_COLNAME | The name of the column specifying the document type |
| FULLTEXT\_BLOBTP\_COLID | The ID of the document type column (from the table itself) |
| FULLTEXT\_LANGUAGE | The language used for the full-text search of the column (expressed as a local identifier) |
| sp\_help\_fulltext\_columns\_cursor **Must be run in the database in which it resides** | |
| USE adventureworks2000  GO  DECLARE @mycursor CURSOR  EXEC sp\_help\_fulltext\_columns\_cursor @mycursor OUTPUT  FETCH NEXT FROM @mycursor  WHILE (@@FETCH\_STATUS <> -1)  BEGIN  FETCH NEXT FROM @mycursor  END  CLOSE @mycursor  DEALLOCATE @mycursor  GO | |
| TABLE\_OWNER | The owner of the table |
| TABLE\_ID | ID of the table |
| TABLE\_NAME | The name of the table |
| FULLTEXT\_COLID | The ID of the column (from the table itself) |
| FULLTEXT\_COLUMN\_NAME | Column of the table that specifies the document type (applicable only when data type is *image*) |
| FULLTEXT\_BLOBTP\_COLNAME | The name of the column specifying the document type |
| FULLTEXT\_BLOBTP\_COLID | The ID of the document type column (from the table itself) |
| FULLTEXT\_LANGUAGE | The language used for the full-text search of the column (expressed as a local identifier) |
| sp\_help\_fulltext\_tables **Must be run in the database in which it resides** | |
| USE adventureworks2000 GO  exec sp\_help\_fulltext\_tables @fulltext\_catalog\_name = 'ctgDocumentSummary' | |
| TABLE\_OWNER | The owner of the table |
| TABLE\_NAME | The name of the table associated with the full-text catalog |
| FULLTEXT\_KEY\_INDEX\_NAME | The name of the unique index in the table |
| FULLTEXT\_KEY\_COLID | The ID of the column associated with the unique index |
| FULLTEXT\_INDEX\_ACTIVE | Specifies whether the columns associated with the full-text index are eligible for queries |
| FULLTEXT\_CATALOG\_NAME | The name of the full-text catalog |
| sp\_help\_fulltext\_tables\_cursor **Must be run in the database in which it resides** | |
| USE AdventureWorks2000  GO  DECLARE @mycursor CURSOR  EXEC sp\_help\_fulltext\_tables\_cursor @mycursor OUTPUT, 'ctgDocumentSummary'  FETCH NEXT FROM @mycursor  WHILE (@@FETCH\_STATUS <> -1)  BEGIN  FETCH NEXT FROM @mycursor  END  CLOSE @mycursor  DEALLOCATE @mycursor  GO | |
| TABLE\_OWNER | The owner of the table |
| TABLE\_NAME | The name of the table associated with the full-text catalog |
| FULLTEXT\_KEY\_INDEX\_NAME | The name of the unique index in the table |
| FULLTEXT\_KEY\_COLID | The ID of the column associated with the unique index |
| FULLTEXT\_INDEX\_ACTIVE | Specifies whether the columns associated with the full-text index are eligible for queries |
| FULLTEXT\_CATALOG\_NAME | The name of the full-text catalog |

# Appendix II – The Application Configuration File

<appSettings>

<add key="CONN\_DMA.DB" value="Data Source=SP6000;Initial Catalog=ECM.Library;Integrated Security=True" />

<add key="CONN\_DMA.DBx" value="Data Source=68.255.7.41;Initial Catalog=ECM.Library;Integrated Security=True" />

<add key="CONN\_DMA.DBy" value="Data Source=SONYLT;Initial Catalog=DMA.UD;Integrated Security=True" />

<add key="CONN\_DMA.DBz" value="Data Source=SP6000;Initial Catalog=DMA.UD;Integrated Security=True" />

<add key="CONN\_RECON\_USERx" value="Data Source=SP6000;Initial Catalog=ReconUserDB;Integrated Security=True" />

<add key="CONN\_RECON\_USER" value="Data Source=SONYLT;Initial Catalog=ReconUserDB;Persist Security Info=True;User ID=sa;Password=junebug" />

<add key="ClientSettingsProvider.ServiceUri" value="" />

<add key="EmailFolder1" value="Personal Folders" />

<add key="SmtpHost" value="smtp.ATT.yahoo.com" />

<add key="SmtpUsername" value="millersuzuki@sbcglobal.net" />

<add key="SmtpPassword" value="suzukiee" />

<add key="dDebug" value="true" />

<add key="CaptureSql" value="true" />

<add key="CaptureSqlDir" value="c:\temp\ECMLib.Log.sql" />

<add key="debug\_frmDocSearch" value="1" />

<add key="debug\_frmQuickSearch" value="1" />

<add key="debug\_frmEmailSearch" value="1" />

</appSettings>