



# The Clarion Handy Tools Page

## CHT Query Language An Overview

# CHT Query Language

*More About Clarion Handy Tools*

## CHT Query Language Syntax

We've been asked to review syntax for CHT Query Language - which some CHT subscribers have come to call GuSQL, pronounced *Gus QL*. Portions of previous help files, like the one pictured below, have covered this in the past. And from time to time I've discussed features added later, and even solicited input from you here in this forum.

But there hasn't been a recent all-in-one place review of CHT Query Language Syntax lately, so that is the purpose of this posting.

### [The Purpose Of CHT Query Language](#)

The point of CHT Query Language is to provide a real-language style method of constructing queries, something that as much as possible resembles the end-user's native language.

Ask developers unfamiliar with CHT Query Language and CHT Query Builder what type of query-building assistance mechanism they prefer and they'll invariably tell you Query By Example. You know Query By Example. That's the query window with 100 different controls on it users are supposed to understand at a glance. It looks different in every application and regardless how simple the intended query it always looks equally imposing.

Ask the same group of developers (hooked on Query By Example) what part of their app generates the largest number of user support requests and they'll tell you it's the querying part. That doesn't happen - at least not to the same extent - with a well thought out CHT Query Language implementation where column headers and query keywords have been adapted to the user's requirements.

People everywhere already speak and write at least one language. In order to do that they have a pre-programmed language interpreter already available in their heads. Developers can leverage that existing skill with CHT Query Language. At the same time by acknowledging that basic, fairly uncomplex queries are the norm and advanced, complex queries are the exception, CHT developers can get a lot of querying mileage out of CHT Query Language and CHT Query Builder.

Query By Example screens cater more to typing avoidance than clarity. CHT Query Builder is not Query By Example. It's there primarily to help querants construct meaningful query "sentences". But it's also a typing avoidance system as are keyword short forms. At it's heart, CHT Query Language is a more familiar and transparent way of constructing queries than QBE will ever be.

I've always resisted requests to enhance CHT Query Language beyond stringing together some basic query conditions and still do. While some elaborations have been introduced, really complex query construction is not it's purpose. That's what raw SQL is for.

### [CHT Query Language Does Not Have To Use English Words](#)

A real benefit of CHT Query Language is that you can configure it to the native language of your users. Or, you can configure specific keywords to more accurately reflect the spirit of your application. I'll give you an example of this a bit further on.

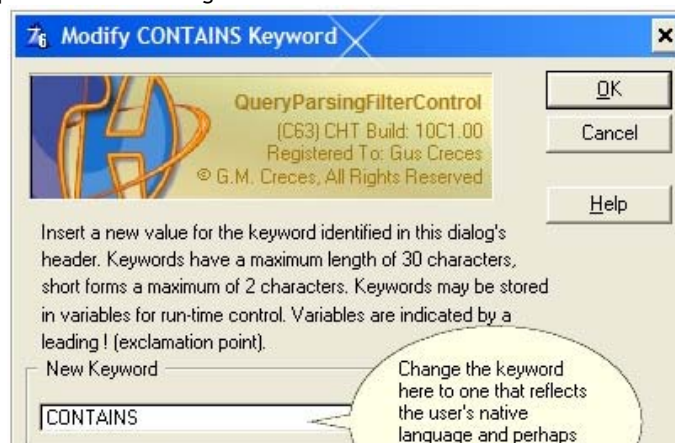
Our query control template (QueryParsingFilterControl) provides a KEYWORDS dialog where translations or modifications can be inserted. The results of any modification can be saved into a .TXT file ( Keywords Definition File) and reused on other instances of the query control in the same application or in other applications. You can, of course, have as many of these "Keyword" definition files as you wish. Be aware that the template looks first for this file in the same directory as your application. The default file name containing the standard set of English-only keywords is "HNDPARSENEW.TXT".

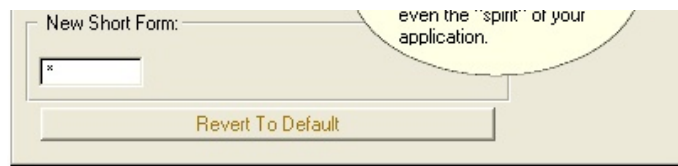
That KEYWORDS dialog looks like this:



If you want to modify the keywords, begin by changing the default Keywords Definition File name to something else, a name that reflects what you're going to do. Then, make your changes and when you're finished, click the SAVE button. All definitions - changed or unchanged - are saved into this new file.

Changes are made by double-clicking on any keyword in the list and substituting a new word (and perhaps a new shortcut) into the dialog provided. That dialog looks like this:





Be careful with short forms. There aren't many keyboard characters that don't appear by themselves somewhere in search text so careless selection of short forms will lead to ambiguity. If you're going to change short forms use a two-letter short form that includes a symbol to produce a combination that you know is never going to appear in search text. For example @C for CONTAINS and @S for STARTSWITH.

An example application that modifies query keywords to match the "spirit" of the application is HNDZINDEX.APP. The two most important things that app points out is:

- Whether a required file is missing from your installation
- Whether a required file is different from the most recently shipped version

Without some changes to the defaults, the queries that would provide this information are:

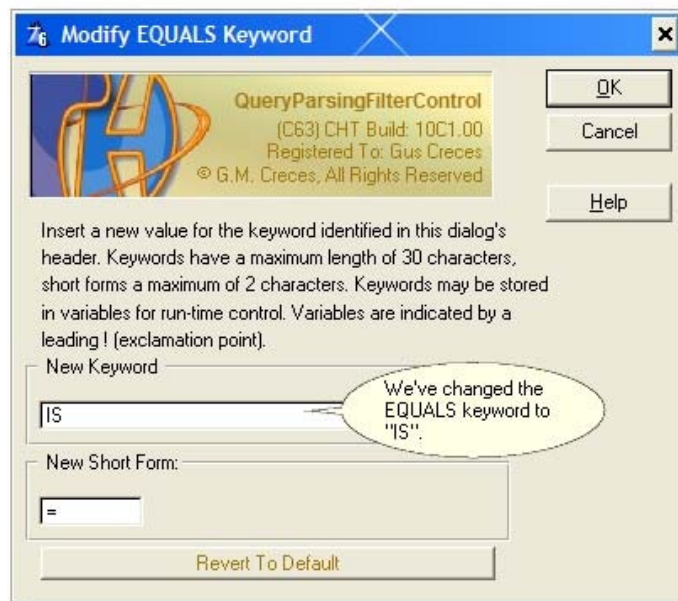
- FILE = 0 / FILE = 1 - The file field is either 1 (the file is there) or 0 (the file is not there).
- STATUS = 0 / STATUS = 1 - The status field is either 1 (your file matches the install version) or 0 (your file does not match the install version)

Our revised versions of these queries are more like "natural" language:

- FILE IS PRESENT / FILE IS MISSING
- STATUS IS CURRENT / STATUS IS OUTDATED

Note that the revised version queries do not need an explanation. The queries themselves say, in each case, what they're looking for with no further requirement to elaborate. That's the idea of natural-language queries.

To affect this change, our application changes the EQUALS keyword to IS. Here is the keyword modification dialog reflecting that change:



That's the only keyword change required for our implementation.

### [Clarion BINDing Tricks That Add Meaning To Queries](#)

So how does PRESENT come to mean 1 and MISSING come to mean 0? And how does CURRENT come to mean 1 and OUTDATED come to mean 0? That's a little Clarion BINDing trick we're playing with the following embed:

```

!These bindings help produce a more "real" language style query, for example: FILE IS MISSING OR
STATUS IS OUTDATED
Missing = 0
BIND('Missing',Missing)
Present = 1
BIND('Present',Present)
Current = 1
BIND('Current',Current)
Outdated = 0
BIND('Outdated',Outdated)

```

Missing, Present, Current and Outdated are variables, local to the procedure declared as follows:

```

Missing BYTE(0)
Present BYTE(1)
Current BYTE(1)
Outdated BYTE(0)

```

### Query Language Can Be Adjusted At Run-Time

Before you ask, the answer is YES. Query syntax can be adjusted at run-time, again because of Clarion's ability to make BINDING adjustments at run-time.

The CHT forum server provides this facility. The query system for this server substitutes variables on the Keywords Dialog instead of constants. To make this work, we need to provide a mechanism that primes these variables from a configuration file. HNDMTSNG.APP does just that. First, take a look at the "Query Keywords" dialog in HNDMTSNG.APP. This dialog can be found on DATA VIEW procedures MessagesJDOView() and MembersJDOView(), since these are the procedures to which message or member queries are sent for data packaging. Note that instead of constant values (hard-coded keywords), this dialog installs variables for all keywords. Variables are prefixed by an exclamation point (!).





The application then primes the keyword variables from the application's configuration file. Remember that all server configuration files are editable by CHT Scripser. New keyword definitions are given to the web client and the web server at the same time by editing the server's configuration file.

```

----- Start: CHT 1001.00 © 2006 (BrowserServer.IDBBuilder) -----
DefaultContainsKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultContainsKeyword ,DefaultContainsKeyword)
DefaultWithoutKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultWithoutKeyword ,DefaultWithoutKeyword)
DefaultStartsWithKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultStartsWithKeyword ,DefaultStartsWithKeyword)
DefaultEndsWithKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultEndsWithKeyword ,DefaultEndsWithKeyword)
DefaultSoundexKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultSoundexKeyword ,DefaultSoundexKeyword)
DefaultEqualKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultEqualKeyword ,DefaultEqualKeyword)
DefaultNotEqualKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotEqualKeyword ,DefaultNotEqualKeyword)
DefaultInKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultInKeyword ,DefaultInKeyword)
DefaultNotInKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotInKeyword ,DefaultNotInKeyword)
DefaultLikeKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultLikeKeyword ,DefaultLikeKeyword)
DefaultNotLikeKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotLikeKeyword ,DefaultNotLikeKeyword)
DefaultBetweenKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultBetweenKeyword ,DefaultBetweenKeyword)
DefaultNotBetweenKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotBetweenKeyword ,DefaultNotBetweenKeyword)
DefaultUnderKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultUnderKeyword ,DefaultUnderKeyword)
DefaultNotUnderKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotUnderKeyword ,DefaultNotUnderKeyword)
DefaultUnderEqualKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultUnderEqualKeyword ,DefaultUnderEqualKeyword)
DefaultNotUnderEqualKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotUnderEqualKeyword ,DefaultNotUnderEqualKeyword)
DefaultOrderByKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultOrderByKeyword ,DefaultOrderByKeyword)
DefaultNotOrderByKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotOrderByKeyword ,DefaultNotOrderByKeyword)
DefaultMonthKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultMonthKeyword ,DefaultMonthKeyword)
DefaultNotMonthKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotMonthKeyword ,DefaultNotMonthKeyword)
DefaultDayKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultDayKeyword ,DefaultDayKeyword)
DefaultNotDayKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotDayKeyword ,DefaultNotDayKeyword)
DefaultYearKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultYearKeyword ,DefaultYearKeyword)
DefaultNotYearKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotYearKeyword ,DefaultNotYearKeyword)
DefaultWordKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultWordKeyword ,DefaultWordKeyword)
DefaultNotWordKeyword = JumpStartGetServerVariables!teem!location(CNH IC:DefaultNotWordKeyword ,DefaultNotWordKeyword)
ConfigQueryRecordCount = JumpStartGetServerVariables!teem!location(CNH IC:ConfigQueryRecordCount ,ConfigQueryRecordCount)
ConfigPurgeRecordMinutes = JumpStartGetServerVariables!teem!location(CNH IC:ConfigPurgeRecordMinutes ,ConfigPurgeRecordMinutes)

```

On-the-fly, keyword definitions are looked up in the application's configuration file.

The server's parsing engine (HNDPARSE) is told about all definition changes and it BINDS these definitions for correct interpretation by the parser. A clever, powerful and very flexible design, even if I do have to say so myself.

```

!----- BEGIN:Query language customizations from HNDNLSV.TXT -----
SELF.SetContains(' * ',DEFAULTCONTAINSKEYWORD) ! (1)
SELF.SetWithout(' - ',DEFAULTWITHOUTKEYWORD) ! (2)
SELF.SetStartsWith(' ^ ',DEFAULTSTARTSWITHKEYWORD) ! (3)
SELF.SetSoundex(' ? ',DEFAULTSOUNDEXKEYWORD) ! (4)
SELF.SetEndsWith(' . ',DEFAULTENDSWITHKEYWORD) ! (5)
SELF.SetEquals(' = ',DEFAULTEQUALKEYWORD) ! (6)
SELF.SetWord(' ' ,DEFAULTWORDKEYWORD) ! (7)
SELF.SetNotEqual(' <<< ',DEFAULTNOTEQUALKEYWORD) ! (8)
SELF.SetNot(' ~ ',DEFAULTNOTKEYWORD) ! (9)
SELF.SetAnd(' & ',DEFAULTANDKEYWORD) ! (10)
SELF.SetOr(' | ',DEFAULTORKEYWORD) ! (11)
SELF.SetGetDate(' @D ',DEFAULTGETDATEKEYWORD) ! (12)
SELF.SetRange(' > ',DEFAULTRANGEKEYWORD) ! (13)
SELF.SetOver(' >> ',DEFAULTOVERKEYWORD) ! (14)
SELF.SetUnder(' <<< ',DEFAULTUNDERKEYWORD) ! (15)
SELF.SetUnderEqual(' <= ',DEFAULTUNDEREQUALKEYWORD) ! (16)
SELF.SetOverEqual(' >= ',DEFAULTTOVEREQUALKEYWORD) ! (17)
SELF.SetMonthComponent(' @M ',DEFAULTMONTHKEYWORD) ! (18)
SELF.SetDayComponent(' @D ',DEFAULTDAYKEYWORD) ! (19)
SELF.SetYearComponent(' @Y ',DEFAULTYEARKEYWORD) ! (20)
SELF.SetORDERBY(' @O ',DEFAULTORDERBYKEYWORD) ! (21)
!----- END:Query language customizations -----

```

On-the-fly, keyword definitions are passed to the CHT query parser for BINDING from the variables containing new keywords. Short forms in this case were not modified.

CHT Real Language Queries And Macro Expansion

HNDMTSNG.APP takes added meaning and avoiding having to type too much, very seriously. To that end, a set of dynamically adjustable MACROS were added to this app to help users in this regard. When you enter the forum by autologging with CHT's Hybrid Web Client, for example, the default query DATE RANGE THISWEEK is applied automatically to provide a list of the most recently posted messages. THISWEEK is a MACRO that's defined in query code as TODAY()-7, TODAY(). So that the query expands inside the server to DATE RANGE TODAY()-7, TODAY(). Similarly, DATE RANGE NOW expands to DATE RANGE TODAY(), TODAY(), and so on.

A CHT query parser (HNDParse Class) function called ReplaceQueryMacro() replaces our MACRO definitions like THISWEEK or NOW with their real meaning. Once again, we've cross-defined THISWEEK, NOW, THISMONTH and others in the server configuration file (look it up in HNDMTSNGCFG.TPS if you don't believe me). The server looks these up and uses ReplaceQueryMacro() to insert the expanded macro back into the user's query before passing the query to the data table. In this case our data tables are .TPS files and CHT parser translates everything to Clarion query language. This would all work equally well with SQL, in which case CHT parser translates everything to SQL before passing the query to the back end.

## CHT Query Language - Word Order

Think of a simple CHT query as a "sentence" or as an "independent clause" or a "syntactic unit" consisting of a Column Name (Subject) a Search Keyword (Verb) and a Search Value (Object). A number of these may be joined together using AND or OR (Conjunction). The only separator used is a space so it is very important to get the word order right and to leave a space between the various elements of the "sentence".

Negation may be introduced using NOT placed **before** the keyword and **after** the column name. Where the equality operator is implied, you may replace NOT EQUAL with simply NOT.

When the RANGE keyword is used, the following VALUE must be a LOW, HIGH pair separated by a comma. RANGE works equally well with numeric and string values.

Date queries may involve also the functions DAY(), MONTH(), YEAR() in which case the column name is inserted between the parenthesis as in DAY(BIRTHDATE) and MONTH(BIRTHDATE). QB1 and QB2 both provide assistance with these functions by inserting the column name into the function for you.

The WORD keyword is a special variation on the CONTAINS keyword. It requires the entire VALUE be contained in the string in such a way that it is preceded and followed by a space, just as a WORD in a sentence has a space in front of it and a space after it. CONTAINS doesn't care whether the value contained meets those conditions.

Parenthesis are not used in CHT Query Language except in one situation where a mixture of AND and OR in a query are likely to lead to ambiguity or an overly lengthy query. Parenthesis impart the meaning GIVEN something or other THEN something OR something OR something else. The GIVEN component is separated from the query inside parenthesis always at the front (left side) of the query. This GIVEN component is then distributed into the final query to all syntactic units in the remaining portion of the query separated by OR.

For example:

(CITY = LONDON) NAME = SMITH OR NAME = JONES

This is equivalent to writing:

CITY = LONDON AND NAME = SMITH OR CITY = LONDON AND NAME = JONES

(DATE RANGE THISWEEK) SUBJECT CONTAINS SMTP OR BODY CONTAINS SMTP

This is equivalent to writing:

DATE RANGE THISWEEK AND SUBJECT CONTAINS SMTP OR DATE RANGE THISWEEK AND BODY CONTAINS SMTP

## Some Abstract Query Examples

COLUMN1 KEYWORD1 VALUE1 AND COLUMN2 KEYWORD2 VALUE2

COLUMN1 KEYWORD1 VALUE1 OR COLUMN2 KEYWORD2 VALUE2

COLUMN1 NOT VALUE1 (implies not equal)

COLUMN1 NOT KEYWORD1 VALUE1 (reverses the meaning of the verb KEYWORD1)

## Some Real World Query Examples

LASTNAME IS SMITH AND CITY IS LONDON

Here, the EQUALS operator has been changed to the word IS.

BIRTHDATE RANGE 6/30/1966, 6/30/2006 AND MONTH(BIRTHDATE) = NOVEMBER

A bound variable named NOVEMBER has been set to the value 11

LASTNAME NOT SMITH

Here we're getting the names of everyone EXCEPT those with the last name SMITH.

FILE IS MISSING or STATUS IS OUTDATED

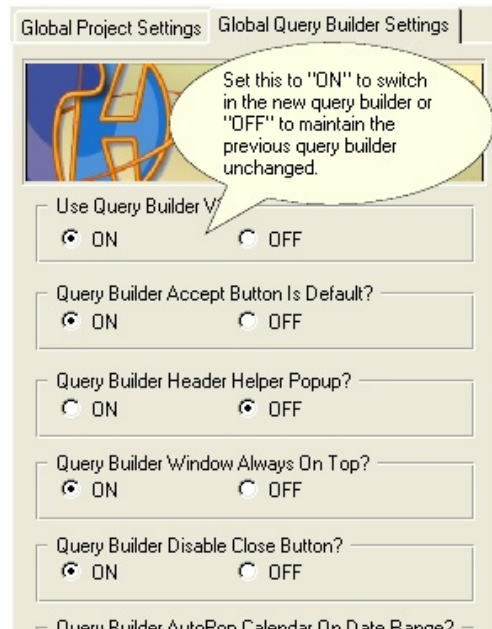
As used in HNDZINDEX.APP to indicate missing or outdated installation files.

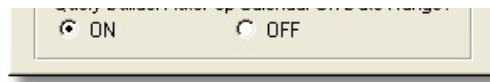
With a bit of imagination and some creative BINDing it's amazing how much CHI queries can be made to resemble real language. And since real language imparts meaning intrinsically, the more your queries can be made to resemble real language the less query support you're going to have to offer.

### [Query Builder 2 Improvements And Options](#)

Build 10C1.0 introduces an optionally switched-in/switched-out query builder with a new look and some selectable features that can be globally enabled on the AACHTControlPanel template. Here is the new query builder window.

We'll review the new features a bit further on in this article. First, if you want to use (or perhaps not use) QB2 you can do so on an application-global basis.





The latest query builder version has some optional behaviors that can be globally determined on the AACHTControlPanel template in the same place you can choose to opt-in or opt-out of QB2.

**Option 1: Query Builder Accept Button Is Default?**

The previous query builder set the CANCEL button as the default button - the one that responds to users striking the enter key when the QB window has focus. With this option you can choose to have QB2 designate the ACCEPT button as default.

**Option 2: Query Builder Header Helper Popup?**

When a browse column header differs from the query word for that column because:

- You have used the "Header Override" feature on our template
- You have included spaces or illegal characters in your column header name
- Your column header name is less than 3 characters

In these cases, QB1 provides a pop-up indicator to make users aware that the query word they've just selected represents a particular column. To make the popup go away you had to modify or fix your column header. QB2 maintains this behavior, but you can optionally switch it OFF here.

**Option 3: Query Builder Always On Top**

QB2 lets you add an "Always On Top" characteristic to the Query Builder window in case any badly-behaving application window wants to hide or cover the query builder window. This makes the window modal and moves it to the top of the window Z-order, keeping it on top until QB2 is closed.

**Option 4: Query Builder Disable Close Button?**

QB1 disables the Windows close button (Red X at top-right of window). You could only exit QB1 via CANCEL or ACCEPT. Here you can opt for an enabled Windows close button. Using the Windows close button is, as expected, equivalent to the CANCEL button.

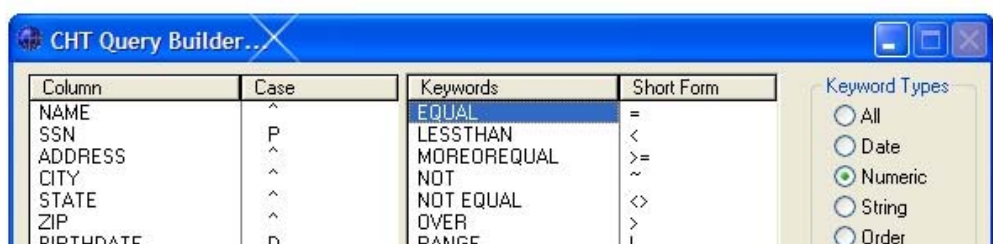
**Option 5: Query Builder Auto-Popup Calendar On Date-Range**

QB2 will help you speed up date-related queries (especially date ranges) by automatically providing a pop-up calendar whenever a date-related column is the active selection. Choose "ON" to enable this feature. QB2, by default also inserts the comma between date range query dates should you forget. For example: BIRTHDATE RANGE 22/9/1962, 25/9/2006.

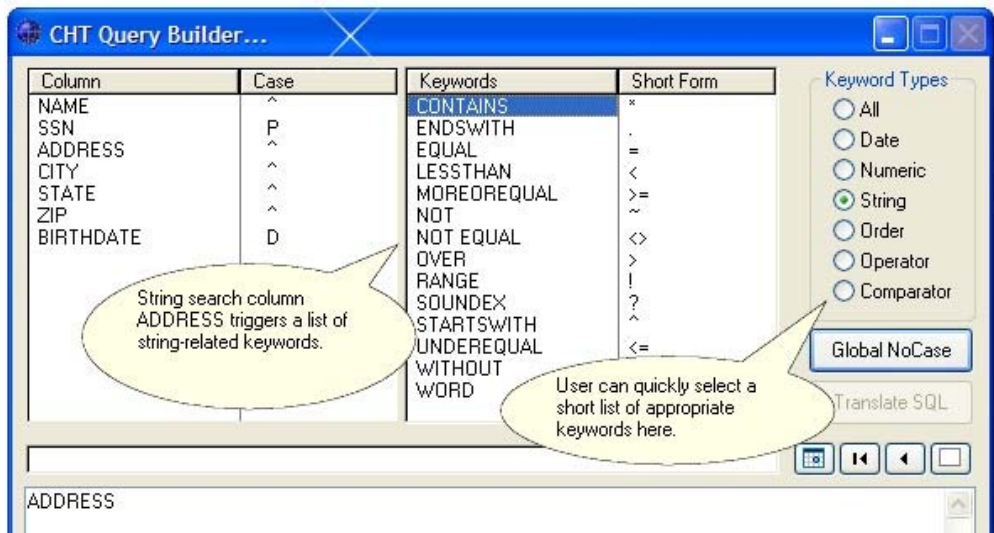
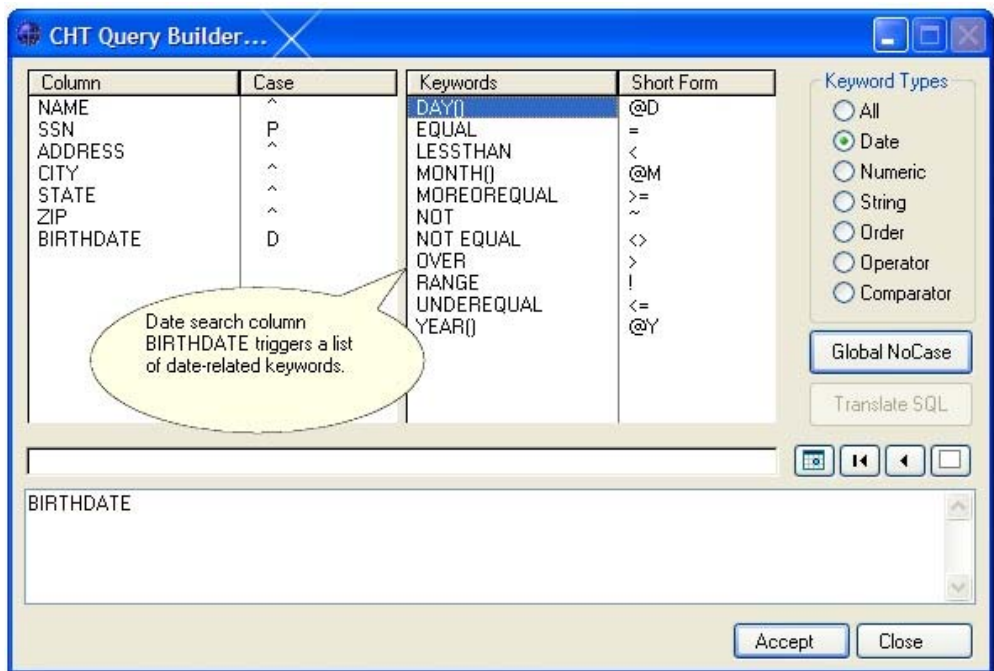
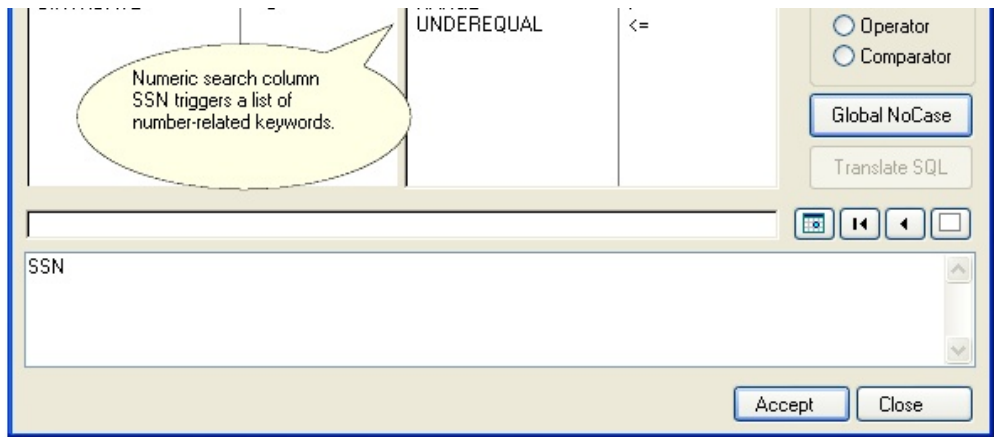
[Query Builder 2 Tries Harder To Be Helpful](#)

Falling well short of reading users' minds, getting them a beer and fluffing their pillows, this query builder tries harder to assist users with query construction by short-listing the keyword list based on the search column selected. Choose a numeric field and only keywords appropriate for numerics are provided. Choose a date field and only date-related keywords are listed. Complete any column query sequence and operators are short-listed in the keywords column. QB2 also lets users manually select for keyword-type preferences without having to scroll through the list of all keywords.

The next three images illustrate the behavior of the KEYWORD column when search columns of different data type are selected from the COLUMN list. Note too, that users can more quickly navigate to keywords of a certain type using the radio buttons provided on the right-hand side of the QB2 window.

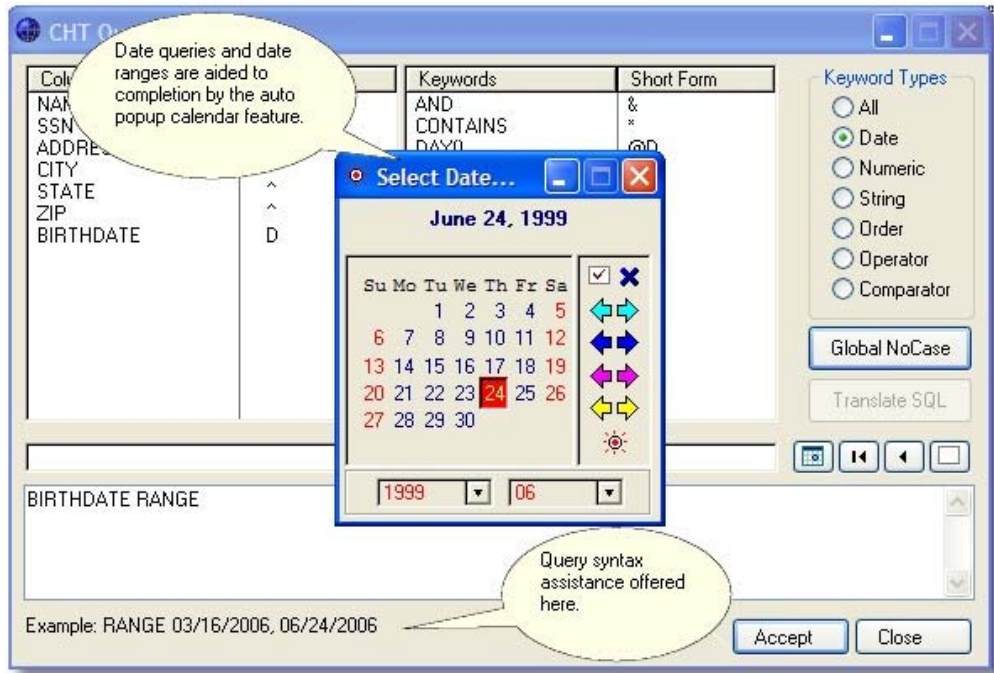




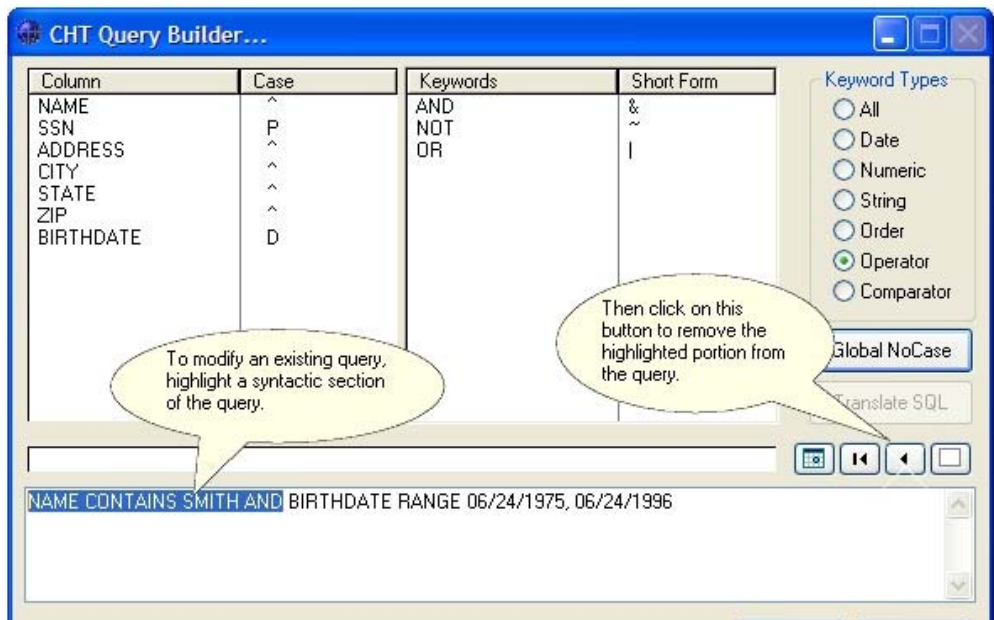




Date queries and date RANGE queries are faster when QB2 is configured to auto-pop the calendar. Syntax assistance is provided in the form of an example where difficulty is likely to be encountered.



Query Builder 1 allowed you to modify exiting queries only from right to left, removing items one by one from the right - from the back of the query to the front of the query. QB2 lets you highlight and remove syntactic sections from any part of the query. See illustration below:



Accept Close

## Other QB2 Features Under Consideration

We've got a list of other features being considered for optional inclusion in QB2. A fairly obvious one is to highlight a syntactic unit in an existing query to replace it with a new one. Another possibility is a dropdown list of previous queries right inside Query Builder. There are still others under consideration. If you have suggestions, feel free to make them.

## Browse Subsorts For ExplorerBrowse And LocatorOverideControl

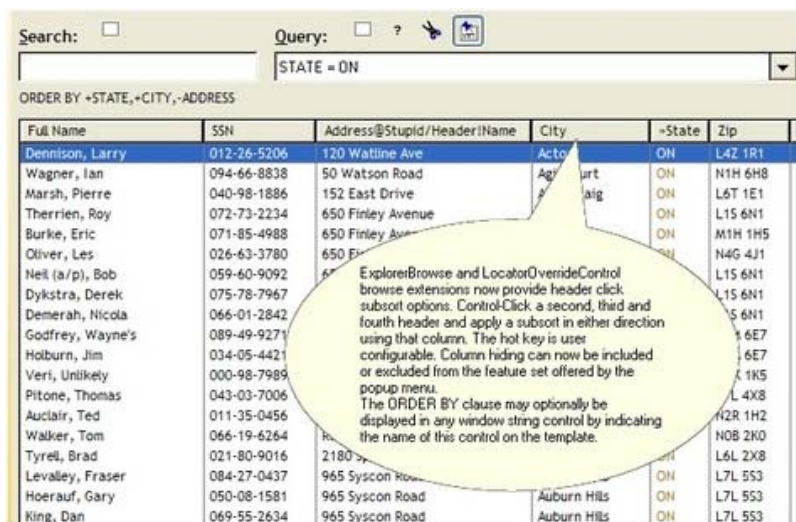
It was brought to my attention a number of times in the last few months that some of you were turning on the new ABC feature called "Sort Headers", even with CHT browse extensions in place. That seems a little illogical to me since our browse extensions already provided sort-header capability and have done so since Clarion 4.



However, it turns out that CHT users were selecting this new ABC feature in order to get header-click-initiated browse subsorts. Unfortunately, in addition to being somewhat redundant, ABC "Sort Headers" clashes with CHT's header sort implementation and acutally breaks some of our browse features.

Although you may not have used it in the past, CHT browses have had browse subsort capability for a number of years. With our query control in place, you could enter an ORDER BY clause, either by itself or in conjunction with a query to rearrange sort order dynamically. And since the query control "remembers" what you enter, it's easy to reselect a specific sort order and impose it on the browse rather than having to retype it or having to click through multiple headers. And of course, CHT queries and ORDER BY statements can be passed from the browse to a report or a process using other templates we provide.

In Build 10C1.0, I've taken the step of unconditionally disabling ABC Sort Headers when our template is installed. That gets rid of the potential conflict in the event you have sort headers enabled. To compensate for the lack of click-header-subsorts in CHT browse extensions I've introduced our version of this feature. With a Control-Click - this keystroke is template configurable - on the browse header a menu pops up. Added to this popup menu now is a +Subsort, -Subsort, Reverse SubSort, and Clear Subsort option.

This morning (Saturday June 24, 2006) I've added a modification to this that lets you not only choose +Subsort and -Subsort. You're also given the capability of reversing any individual sort element in the ORDER BY clause. For example, in an ORDER BY +STATE,+CITY,+ADDRESS order clause, you can revisit the CITY column and reverse it only, resulting in ORDER BY +STATE, -CITY, +ADDRESS. You can of course, also reverse the entire sort order using the "Reverse Subsort" menu item.



Search:  Query:  ?  

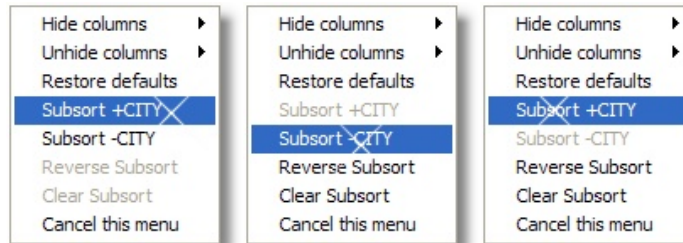
STATE = ON

ORDER BY +STATE,+CITY,-ADDRESS

Full Name	SSN	Address@Stupid/Header!Name	City	-State	Zip
Dennison, Larry	012-26-5206	120 Watline Ave	Acton	OH	L4Z 1R1
Wagner, Ian	094-66-8838	50 Watson Road	Agar	OH	N1H 6H8
Marsh, Pierre	040-98-1886	152 East Drive	Agar	OH	L6T 1E1
Therrien, Roy	072-73-2234	650 Finley Avenue	Agar	OH	L15 6N1
Burke, Eric	071-85-4988	650 Finley Avenue	Agar	OH	M1H 1H5
Oliver, Les	026-63-3780	650 Finley Avenue	Agar	OH	N4G 4J1
Neil (a/p), Bob	059-60-9092	650 Finley Avenue	Agar	OH	L15 6N1
Dykstra, Derek	075-78-7967	650 Finley Avenue	Agar	OH	L15 6N1
Demerah, Nicola	066-01-2842	650 Finley Avenue	Agar	OH	L15 6N1
Godfrey, Wayne's	089-49-9271	650 Finley Avenue	Agar	OH	L15 6E7
Holburn, Jim	034-05-4421	650 Finley Avenue	Agar	OH	L15 6E7
Veri, Unlikely	000-98-7989	650 Finley Avenue	Agar	OH	L15 6N1
Pitone, Thomas	043-03-7006	650 Finley Avenue	Agar	OH	L4X8
Auclair, Ted	011-35-0456	650 Finley Avenue	Agar	OH	N2R 1H2
Walker, Tom	066-19-6264	650 Finley Avenue	Agar	OH	N0B 2K0
Tyrell, Brad	021-80-9016	2180	Agar	OH	L6L 2X8
Levalley, Fraser	084-27-0437	965 Syscon Road	Auburn Hills	OH	L7L 553
Hoerauf, Gary	050-08-1581	965 Syscon Road	Auburn Hills	OH	L7L 553
King, Dan	069-55-2634	965 Syscon Road	Auburn Hills	OH	L7L 553

ExplorerBrowse and LocatorOverideControl browse extensions now provide header click subsort options. Control-Click a second, third and fourth header and apply a subsort in either direction using that column. The hot key is user configurable. Column hiding can now be included or excluded from the feature set offered by the popup menu. The ORDER BY clause may optionally be displayed in any window string control by indicating the name of this control on the template.

In the next example, using HND2.APP, we've begun by performing a single left mouse click on the STATE column to place the browse data in STATE order. The next two images illustrate what you would see after a Control-Click on the CITY column. The first Control-Click shows us the option to sub-sort by +CITY or -CITY. We selected to impose a +CITY sort order added to the already-present STATE order. A second Control-Click on the CITY column now offers us only -CITY. Were we to select this option, the secondary sort is reversed and a third Control-Click on the same CITY header now offers only a +CITY option. As soon as any sub-sort is added to the browse two other menu items - Reverse Subsort and Clear Subsort - also become available.



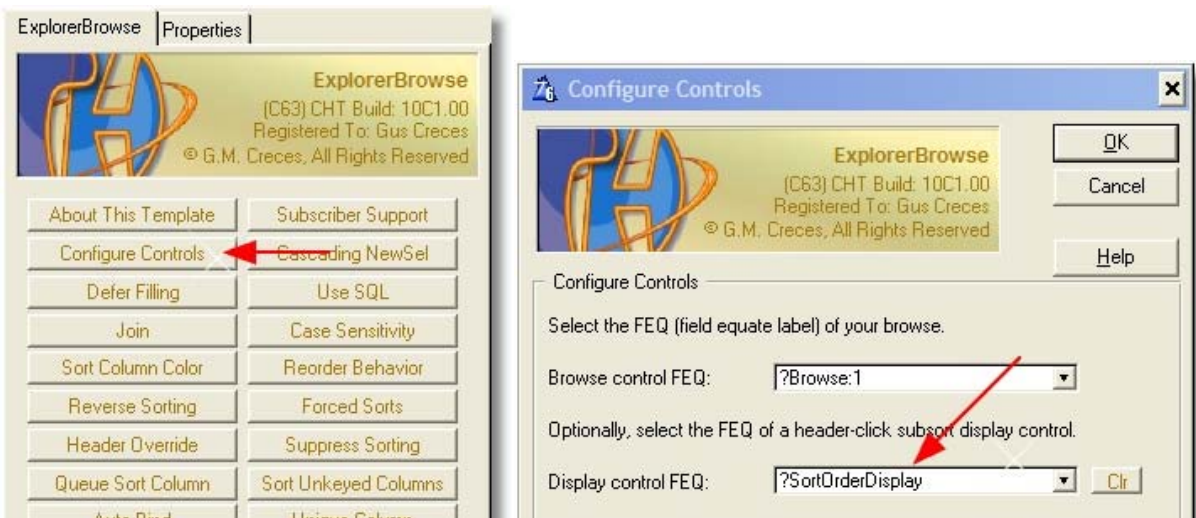
The next image, illustrates the result of a 4 level sub-sort: ORDER BY +STATE,+CITY,+ADDRESS,+NAME. An optional, user-installed, string control may be designated on the template to dynamically display the ORDER BY statement as subsorts are imposed or cleared.

ORDER BY +STATE,+CITY,+ADDRESS,+NAME

Full Name	SSN	Address	City	State
Godfrey, Wayne's	089-49-9271	27 Hamilton Road	Ajax	ON
Holburn, Jim	034-05-4421	27 Hamilton Road	Ajax	ON
Burke, Eric	071-85-4988	650 Finley Avenue	Ajax	ON
Demerah, Nicola	066-01-2842	650 Finley Avenue	Ajax	ON
Dykstra, Derek	075-78-7967	650 Finley Avenue	Ajax	ON
Neil (a/p), Bob	059-60-9092	650 Finley Avenue	Ajax	ON
Oliver, Les	026-63-3780	650 Finley Avenue	Ajax	ON
Therrien, Roy	072-73-2234	650 Finley Avenue	Ajax	ON
Veri, Unlikely	000-98-7989	32 Nottawasaga Road	Akron	ON
Pitone, Thomas	043-03-7006	919 Fraser Drive	Allanburg	ON

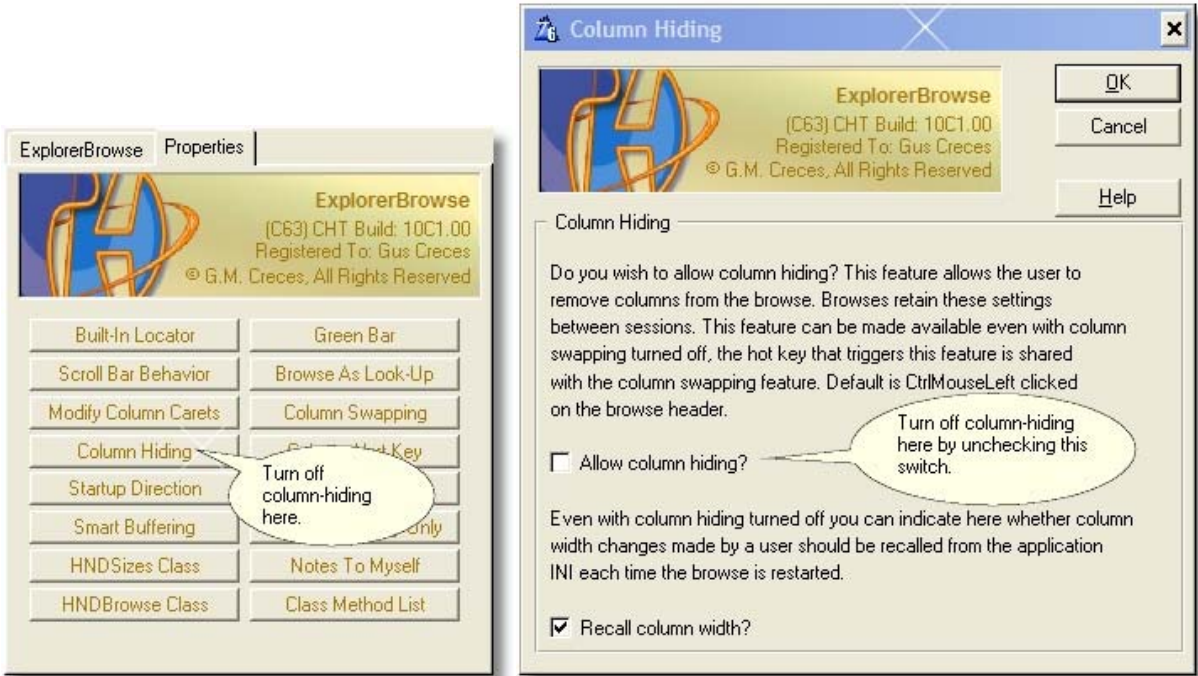
A window string control can optionally display the browse ORDER BY clause.

The ORDER BY display control is dropped on the window as you would any ordinary string control. Once dropped, navigate to these dialogs to tell our template that you want this control to begin displaying ORDER BY information.

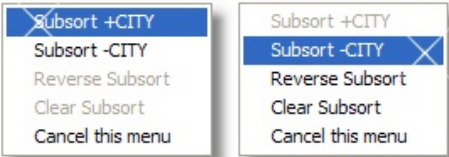




As of today, you have the option of disabling column hiding and still having the subsort option in place. The next two images illustrate how to do that.



Once column-hiding is disabled, the subsort menus appear as follows:



That's all there's to it. Have fun with this new CHT feature. ;-)