

Icetips Cowboy SQL Templates

SQL

User's Guide

Template Reference

Class Reference



The Power of SQL at your fingertips!

Version 6.000

Table of contents

TABLE OF CONTENTS	2
WELCOME	7
INTRODUCTION	8
DEMO APPLICATIONS	9
INSTALLATION	10
UPGRADING FROM EARLIER VERSIONS	11
ADDING THE SQL TEMPLATES FOR THE FIRST TIME	12
REGISTERING THE TEMPLATE	12
CREATING A SQL BROWSE	12
USING A REGULAR WINDOW PROCEDURE	12
USING THE PROCEDURE TEMPLATE	13
CONTROL TEMPLATES	14
ICETIPS SQL BROWSE	14
MAIN WINDOW	14
BROWSE BOX BEHAVIOR	15
General	15
Alertkey	15
Column swapping	15
Save browse format	15
Use ABC Toolbar	16
Greenbar	16
Default Sort	17
Record Tagging	17
Visual Indicators	17
ClarioNet Alertkey	18
Filter	18
Global filter/range limit	18
Column filter/range limit	18
Filter scope setting	19
Sort	20

Sort column color	20
Forced sorts	20
Suppress sorting	21
Join	21
Synchronize child	21
Synchronize Multi child	21
WHERE clause	22
Force INNER	23
Data Access	23
Hot fields	23
Fill on demand	23
Stored procedures	23
Function fields	24
Extended SQL	24
After SELECT before FROM	24
After FROM before WHERE	24
After WHERE before ORDER BY	25
After ORDER BY	25
Replace Everything	25
Replace ORDER BY	25
Fetch on select	25
Smart buffering	25
Page size	25
Pages behind	25
Pages ahead	26
Timeout	26
Reset Fields	26
Active Invisible	26
Variables	26
Force Use of the - primary- file and related files	27
Colors	28
Icons	28
Styles	28
Tooltips	29
Classes	29
ICETIPS SOFTWARE ONLINE SUPPORT I-SOS	29

ICETIPS SQL BROWSE LOCATOR AND CCS SQL BROWSE LOCATOR	30
SHARED LOCATOR	30
PROGRESSIVE LOCATOR	30
DATES IN LOCATORS	30
ICETIPS SQL UPDATE BUTTONS	32
UPDATE PROCEDURE	32
Selecting update procedure	32
Popup menu	32
Hotkeys	32
INSERT OPTIONS	33
Locate and Isolate	33
Suppress Clear	33
Field priming	34
ICETIPS SQL SELECT BUTTON	35
ICETIPS SQL CANCEL BUTTON	35
ICETIPS SQL CLOSE BUTTON	35
DROP LIST OF SORT ORDERS	36
GLOBAL EXTENSION TEMPLATES	37
GET USER NAME FROM NETWORK	37
USE DEBUGGING CLASSES	37
GLOBAL OPTIONS	37
PROCEDURE EXTENSION TEMPLATES	38
NULL FIELDS BEFORE ADDING RECORD	38
STORE SELECT STATEMENT	38
WEB SIZE EXTENSION	38
SECURITY PAGE SETUP	38
ROUTINE DECLARATION	38
CALCULATE MONTHLY LOAN PAYMENT	38
RESTORE CHILD AFTER CANCEL	38
CODE AND WORKING EXAMPLES	39
RESET BROWSE BASED ON VALUE FROM A DROPDOWN	39
CHILD BROWSE ON PARENT FORM	40
SYNCHRONIZING MULTIPLE CHILD BROWSES	41

FILTER USING AN OPTION AND RADIO BUTTONS	43
CLASS REFERENCE	44
CCSSQL1 CLASS	44
CLASS PROPERTIES	44
CLASS METHODS	53
CCSBUTTONS CLASS	81
CLASS PROPERTIES	81
CLASS METHODS	83
CCSSIZES CLASS	85
CLASS METHODS	85
CCSLOCMANAGER CLASS	86
CLASS PROPERTIES	86
CLASS METHODS	86
CCSTOOLBARLISTBOXCLASS CLASS	88
CLASS PROPERTIES	88
CLASS METHODS	88
COMPATIBILITY AND TECHNICAL ISSUES	89
TECHNICAL SUPPORT	90
EMAIL	90
NEWSGROUPS	90
INTERNET BULLETIN BOARD	90
ICETIPS SOFTWARE ONLINE SUPPORT, I-SOS	90
INSTALLED FILES	91
LAST MINUTE CHANGES	92
DEFAULT SORT ORDER	92
SORT ORDER	92
LIMITATIONS	92
CHANGES FROM PREVIOUS VERSION	93
VERSION 6.000 FINAL RELEASE	93
VERSION 6.000 BETA B, MAY 5, 2003	95
KNOWN ISSUES IN 6.0, BETA B	95
CHANGES FROM BETA A	95

VERSION 6.000 BETA A, FEBRUARY 20, 2003

96

CHANGES FROM 5.5

96

Welcome

Welcome to the Icetips Cowboy SQL templates!

This is our first release of those templates which we acquired from its author Andy Stapleton in July 2002. We have added a lot of new features and improvements in this release.

Please read through this manual as there are so many changes that we have made all over in the templates and while the overall look and feel of the templates hasn't changed, a lot of the internals have changed, or more precisely things have been added.

For your convenience, we have put an image on the left margin where we discuss new or improved features of the SQL templates.

NEW

This image indicates a new item!

This documentation will not teach you how to use SQL, that is outside the scope of this project, but it will teach you how to use the SQL templates and get familiar with working with SQL databases using our templates. We have a special section on our website that is dedicated to SQL and nothing else and it has a lot of SQL related resources and links. Installed with the SQL templates is a PDF file with a series of articles written by Dan Pressnell which we have got his kind permission to use and install with our product. Dan's articles are a goldmine for anyone who wants to explore SQL in depth. He has also written classes that access the ODBC drivers directly using the standard ODBC API.

Introduction

The Icetips Cowboy SQL templates have been around for about 8 years or since 1996. They came out first for Clarion templates and later on for the ABC chain. In July 2002, Icetips Software took the templates over from their author, Andy Stapleton and since then we have been working on various new features and improvements. In this manual, we will usually refer to the Icetips Cowboy SQL templates as the "SQL templates"

The heart of the Icetips SQL templates are a browse control template that replaces ABC browses. This means that in existing applications, the browse procedures need to be re-created using the SQL templates. Also included are Locator template which adds a filtering locator to the browse, update button template which adds the usual update buttons, cancel and close control templates. Together these templates make up the SQL template product, along with several class files, which are detailed in the Class Reference at the end of this manual.

The SQL templates construct SQL statements, based on the developer settings, that are passed on to the Clarion View structure that is used for the browse. This SQL statement construction comes in two flavours, full, which is turned on by checking the "Force Inner" option, and partial, which is the default and is turned on by unchecking the "Force Inner" option. The difference is that with the full construction, the entire **Select** statement is built by the SQL templates and passed on to the Clarion View with **Prop:SQL**. With the partial construction, the **Order by** and **Where** clauses are built and passed to the Clarion View with **Prop:SQLOrder** and **Prop:SQLFilter** respectively. By forcing the inner joins and thus the Prop:SQL construction, the templates can also access SQL functions to retrieve data. This is not enabled if the inner joins are not forced.

NEW

The SQL Update buttons have pretty much the same functionality as in any other browse template - they allow the user to insert, change or delete a record from the database table. In this version the developer can optionally enable Popup menu and alert any hotkeys they want for the buttons as well as define multiple hotkeys for various actions such as insert, change, delete and select. Also new in this version are options to suppress the Clear on the table record done by the Insert method and also do pre- and post-priming of the record. The Pre-priming is done before the record is added to the database. This can be handy to set values on fields that enforce Referential Integrity to other tables, for example lookup tables. The Post-priming is done after the record has been added and in essence overwrites what the Pre-priming did. This can be used to NULL out fields that enforce Referential Integrity and force the user to select a value from lookup tables etc. Both the Pre- and Post-priming allows for a value or variable be used, the field set to NULL or a Clarion function can be called to assign values to the field. These options are only available if Suppress Clear is checked and so is Clear record, which is mandatory to use with Suppress Clear as otherwise you risk have the values of the currently active record be written into the new record!

The SQL Select button is like any other Select button, it selects the currently active row in the browse and the values active in the browse are available where the browse was called from.

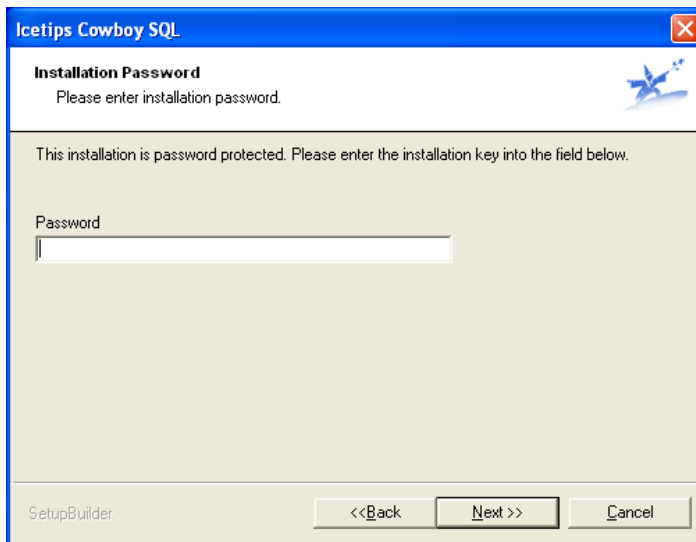
There are TWO almost identical Locator control templates, one is the original Locator control, but the new one is very similar but works independently on multiple browses and can not be shared, i.e. the locator is tied to the browse it locates on. Many users find it confusing to have a shared locator for multiple browses like the original locator did.

Demo Applications

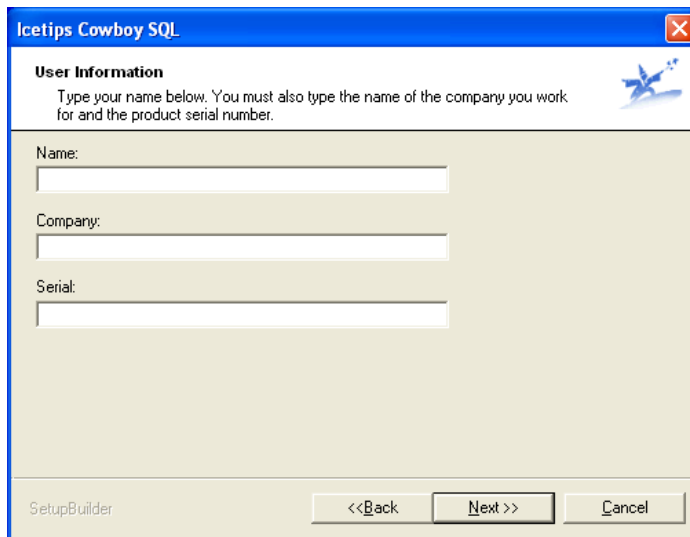
This is currently under construction - may not be distributed with the initial release. In that case please check with us through the Icetips Software Online Support program for updates.

Installation

The Icetips Cowboy SQL templates use an installation program created with LinderSoft's SetupBuilder 4.03. This install is simple to run. It has two layers of authentication using installation password and serial number. Both are required during the install and are supplied via email from our online store when the product is purchased.



Entering Installation password in Installation program



Enter user information in Installation program

If you lose this information, you can always send us an email to support@icetips.com, requesting the installation information or log into the Icetips Software Online Support Center.

When you purchase our software, you will get a user name and generated password which you can use to log into our Support Center. You can query our database for your product information including download and installation information and our Center will send them to you via the email address that you supplied when you purchased.

It is up to you to keep your user profile up to date and once you have logged into our system you can change your password to something more appropriate.

You **MUST** enter the Name, Company and Serial number exactly as they are in your registration email that you will get when you complete the purchase. These fields are case sensitive and if not entered as expected the install will not work.

The installation will attempt to register the template. If it fails, then you need to register the template manually.

At the end of the installation our Post-Install program runs and does some file cleanup and

housekeeping. Among other things it checks the Clarion Redirection file to make sure that paths that are needed by the product are appropriate so that the Clarion environment can find the files, such as template files, images, icons, etc.

Please let us know immediately if you have problems during the install so we can take a look at it and fix it to prevent other users from having problems with the installation.

Upgrading from earlier versions

This new version of the SQL templates is a major upgrade. We have not found any migration problems from the earlier version, but urge customers who are upgrading to take precautions in case they run into some migration issues!

We strongly recommend that you read through this manual to familiarize yourself with the changes and improvements and what impact they may have on your existing applications.

Please read!

Even though we do not expect any problems at all with this new version, we do however recommend that if you are upgrading from an earlier version and you take the following precautions, **before** installing the new version of the SQL templates:

1. Make a backup of applications that use the original version of the SQL templates
2. Make a backup of the original templates and classes - see instructions below

We recommend that you uninstall earlier versions of the Icetips Cowboy SQL templates. If you do not have an uninstall option, then locate the **Ccsabc.tpl** and **Ccsabcex.tpw** files which will probably be in your Template directory, i.e. C:\C55\Template directory if your Clarion 5.5 installation is in C:\C55 directory.

We encourage you to make a backup copy of the old version in case you find some problems with the new one and want to revert to the previous version. Then please also include all CCS*. * files in the LibSrc directory:

```
CCSBUTNS.CLW
CCSTOOLB.CLW
CCSLOCAT.CLW
CCSSQL1.CLW
CCSSIZES.CLW
CCSLOCAT.INC
CCSSQL1.INC
CCSBUTNS.INC
CCSTOOLB.INC
CCSSIZES.INC
```

Put the files in a safe location and then delete them from the LibSrc directory.

The new installation will put the template files into \3rdParty\Template subdirectory from your Clarion 5.5 or Clarion 6.0 installation root directory. The class files (*.clw and *.inc files) will still be placed into the \LibSrc directory as they were in the older version.

If you experience any migration problems, we would appreciate if you would let us know as soon as possible via email to support@icetips.com so we can find the cause and fix the problem.

On page 93 there is a fairly comprehensive list over all of the changes that have been made since version 5.5.

Adding the SQL templates for the first time

Registering the template

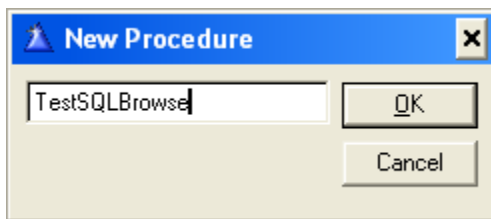
Normally the installation program will register the template for you during the installation. If you choose not to have it do that or the installation program fails in registering the template, you need to register the template manually.

To register the template manually, open the Clarion Integrated Development Environment (IDE) and select "Setup|Template Registry" from the main menu. Then click on the [Register] button and navigate to the installation template directory (i.e. C:\C55\3rdParty\Template or C:\Clarion6\3rdParty\Template) and select the Ccsabc.tpl file. Clarion will now register the template chain and you are ready to start using the template.

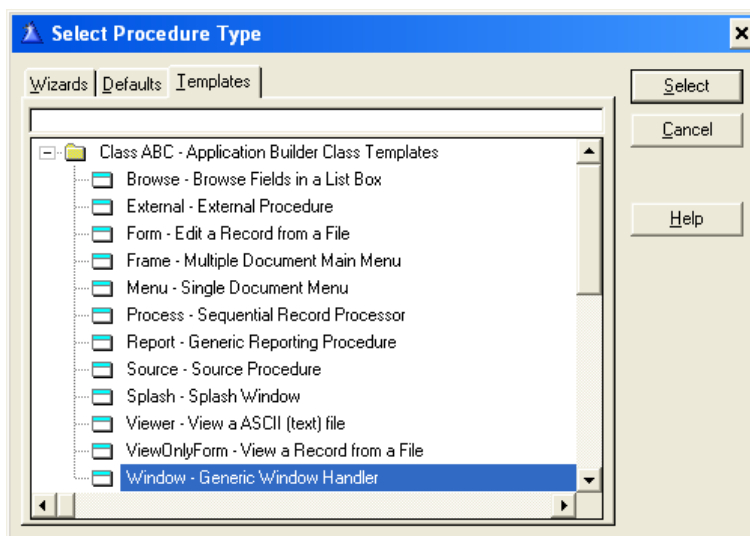
Creating a SQL browse

There are two ways to create a SQL browse procedure. You can either create a window procedure and then drop the SQL browse control on it or you can use one of the SQL Browse procedure templates. Currently there are two such templates, "Icetips SQL Browse with update buttons" and "SQL Only browse with update". The first is a slightly modified version of the second which is the procedure template that was in previous version(s).

Using a regular Window procedure



To create a SQL browse window, you first create a window procedure in Clarion. From the main menu in the Clarion IDE, select "Procedure|New" or hit the Insert key on the keyboard. Once the "New Procedure" window comes up, type in the name for the procedure, in our example we will use TestSQLBrowse as the procedure name. Click the OK button on the "New Procedure" window and the



"Select Procedure Type" window will come up. Select the "Templates" tab, make sure you are not on the "Defaults" tab as it has different entries, and then select the "Window - Generic Window Handler" from the "ABC - Application Builder Class Templates" Class. This will create a completely empty Window procedure.

To create the window so you can start populating the control templates and other controls that need to be there, click on the [Window] button and a "New Structure" window will come up. Select the "MDI

Child Window" entry and click the [OK] button. Now you have a window to work with. Resize the window in the window formatter so that it is about 400 dialog units wide and 200 dialog units tall.

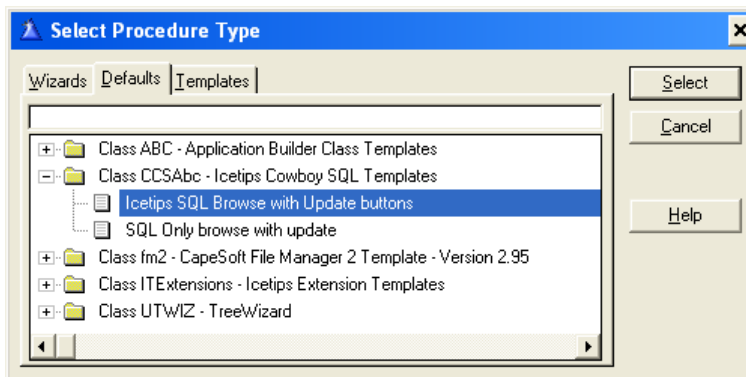
Once the window is created, you can add the browse control template. Click the [Control Template] button or select "Populate|Control Template" from the Window Formatter menu. Then select the SQLBrowseNL from the "CCSAbc - Icetips Cowboy SQL Templates" class and select the upper left corner where you want to drop the listbox control.

By default the listbox control is not populated with any fields from the database. Right click on the listbox control and select "List Box Format..." from the popup menu to open up the listbox formatter. Then add the columns that you want the browse to show.

This is all very similar to adding a regular ABC browse to a window.

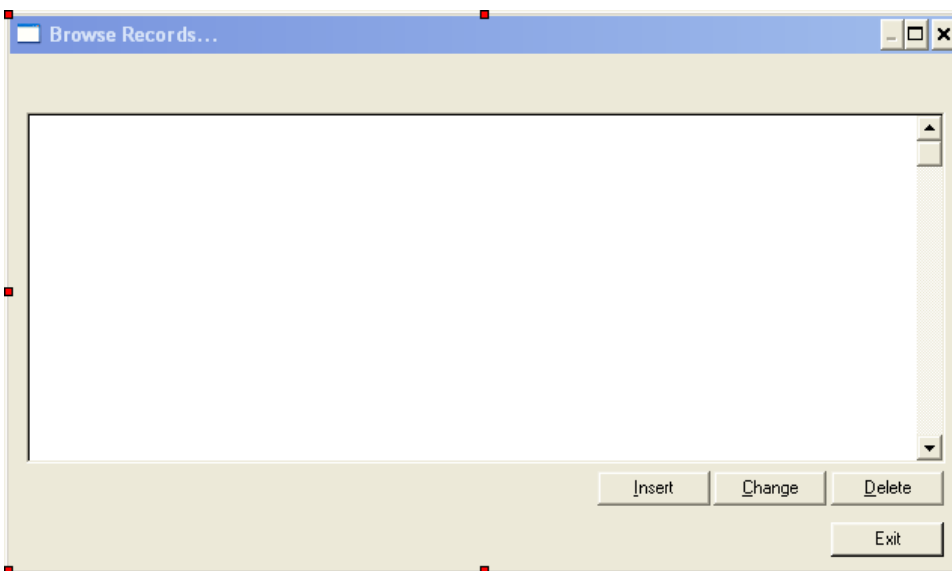
Once the browse is in place, you can add update buttons, select button as well as locator. If you have multiple browses on the same window, you will be prompted for which listbox you want to add the additional control templates, same as if you have multiple ABC browses on the same window.

Using the Procedure template



Selecting the Icetips SQL Browse procedure template

There are two very similar procedure templates provided with this version of the SQL templates. One includes update buttons and select button and is identical to the original procedure template, but the other includes update buttons only and is slightly modified version of the original SQL procedure template. Try them both and see how they work for you.



The resulting SQL browse procedure

The new procedure is simple, yet contains what needs to be on the browse. Now you can add fields to it and start setting it all up. For more information on the templates, refer to the **Control Template** section.

Control Templates

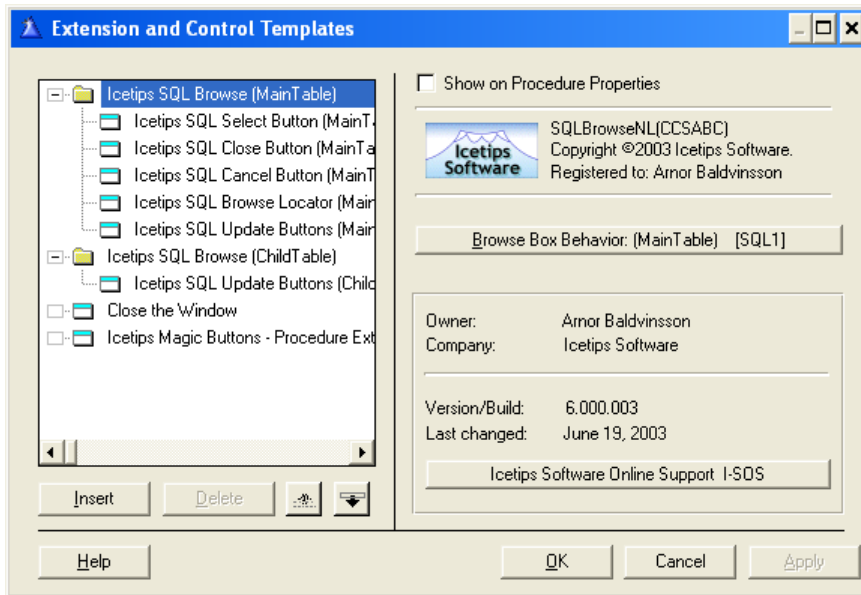
Icetips SQL Browse

The main power of the SQL templates is stored in the "Icetips SQL Browse" control template. This is the foundation for everything else and is required by the other control templates. This is where all the browse functionality is set up including how it structures the Select statement sent to the backend as well as visual appearance settings such as colors, icons and styles.

In the following documentation, we will go through each window, tab and button and explain as well as possible what these functions do and how they will affect the generated SQL or other controls on the window and how other controls will affect the SQL browse.

Main window

The main window for the Icetips SQL Browse control template contains the "Browse Box Behavior" button that opens up a window with several tabs and buttons with all the options available in the control template. The button has information both about the primary table that is used in the listbox as well as the object name that the template uses for the particular browse and listbox.



The main window of the Icetips SQL Browse control template

This makes it easy to see and remember what class label to use when handcoding.

In a box in the lower part of the window the template displays information about the registered owner and the version information, both the version/build number as well as the last date it was changed. The information you see on this window will probably not show the same build/change information as the screenshot, but that is normal.

There is also the "Icetips Software Online Support" button, which will take you to another window which is described in more detail on page 29. If you need help at any point in the templates, use this button to open up the direct options that you have including an option to log into our online support center for help and information.

At the top of this window as well as all other windows in these templates, is a section with our Logo, the name of the template, our copyright notice and the name of the registered owner of this copy of the Icetips SQL.

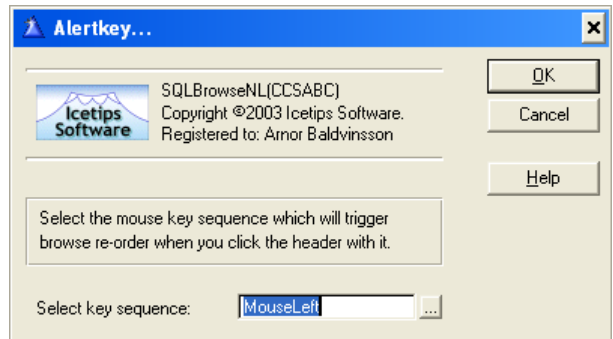
Browse Box Behavior

When you click on this button, another window comes up with a lot of tabs and buttons on each tab. We will go through each tab and each button on those tabs and explain what they do and how they affect the browse box. We have skipped some buttons that only display information and do not interact with the developer.

GENERAL

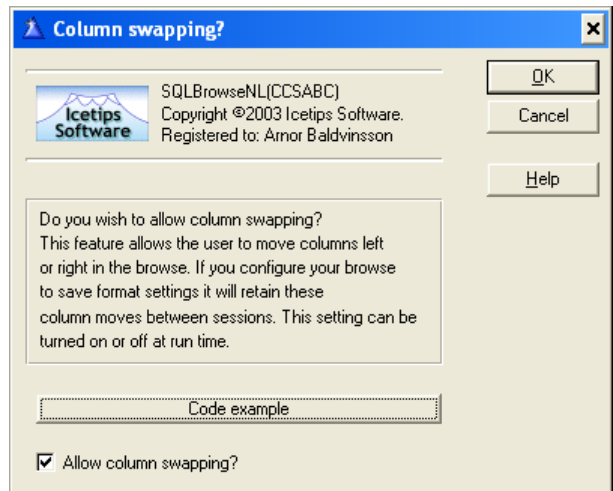
Alertkey

This allows you to specify what key to use to trigger resorting of the browse when you click the header with it. For rather obvious reasons this needs to be some mouse key combination. We have not yet implemented a way to switch the sort orders with the keyboard but hope to have that possibility implemented in the next release of the SQL templates.



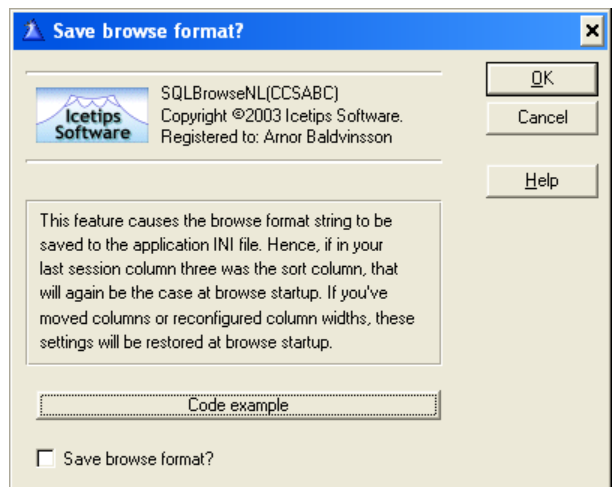
Column swapping

Column swapping allows you to move the columns from left to right or right to left in the listbox. This makes it easy for users to customize the browses. There are no methods to call for this, only a property to set and it can be set at any time if you like. Click on the "Code example" button to see how to turn the column swapping on or off. You can copy/paste the code from the text fields on the Code example window and it will use the correct object name based on the template instance you are working with.



Save browse format

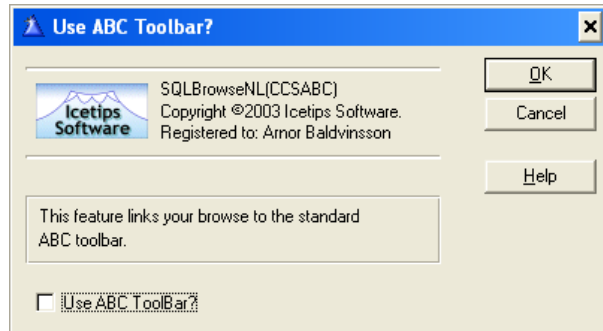
This allows the user to modify the columns in the browse and when the browse window closes, the format is saved and then restored when the user opens the window again. This will also save the active sort column and restore it when the window opens. Note that if the Default sort column is set, it will override the sort column, so that the browse always starts up with the defined sort column. There are no methods to call for this, only a property to set and it can be set at any time if you like. Click on the "Code example" button to see how to turn the Save on or off. You can copy/paste the



code from the text fields on the Code example window and it will use the correct object name based on the template instance you are working with.

Use ABC Toolbar

This allows the browse to use the ABC toolbar for navigation. However, the ABC toolbar class has problems with navigating multiple listboxes on the same window. This makes it less than ideal solution for navigating records in listboxes. We do not recommend that you use this option, but if you only have one listbox on the window or only want one listbox to be navigated by the toolbar, you should not have any problems with this option.



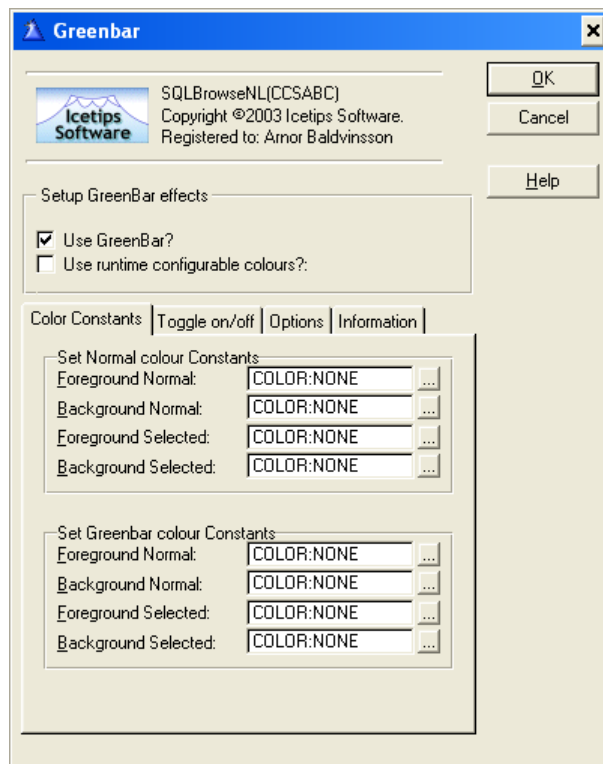
Greenbar

The Greenbar option in the Icetips SQL templates is implemented directly from JaduTech's freeware Greenbar templates. The author, Steve Bottomley of Australia, was so kind as to give us full permission to integrate his template code into ours. It allows you to set up very nice bar effects on your listbox and even if the name suggests green, it can be any color combination you like. You can set the background, foreground, selected background and selected foreground as well as conditional colors and all kinds of other options.

For example try to set the Greenbar Background Normal color to one of the following values:

- 0B7E2ECH - pale brown
- 0CEFDE1H - pale green
- 0D2D6FFH - pale red/pink
- 0FEE0E4H - pale lavender
- 0BBFDFBH - pale yellow
- 0A6BFDH - pale orange/yellow

We are not going to go into further details on this template in this version of the documentation, and the templates are fairly self explanatory. You can visit Steve's website from a button on the information tab. He has several other freeware products and example applications for Clarion.



NEW

Default Sort

This option allows you to specify a column as the default sort column. When you check the "Set Default Sort column" you can select the column that you want to use as the default column. Note that if you at later time remove that column, no column will inherit the default column.

Record Tagging

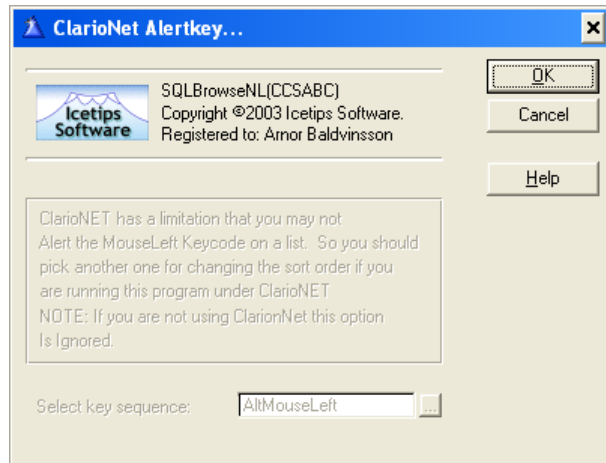
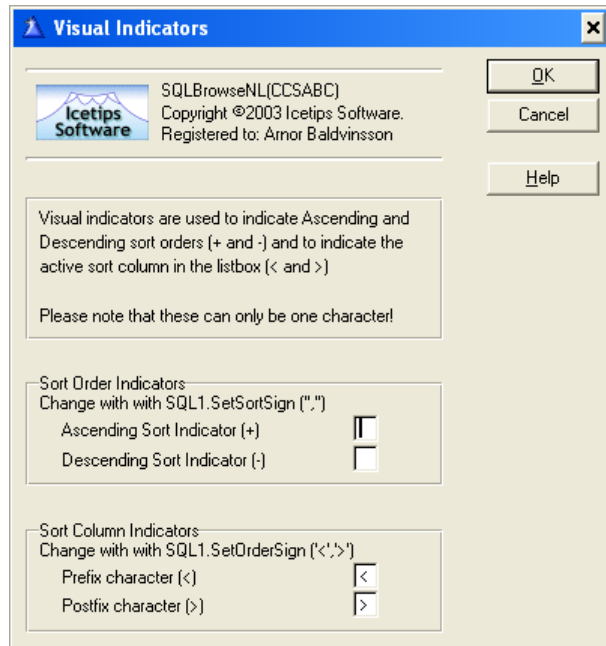
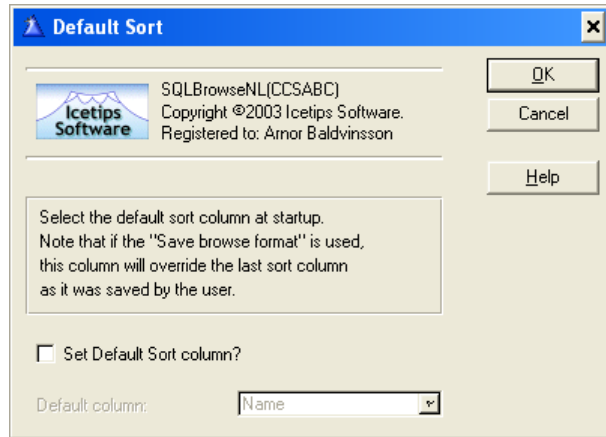
Not implemented in this version. We expect to have record tagging implemented in an interim release in the fall of 2003. Please check back for availability and check our website regularly.

Visual Indicators

The Visual Indicators allow you to change the sort order indicators and the sort column indicators.

The Sort Order is by default indicated with a + or a - in front of the text in the column header. However you may want to use some other characters to indicate ascending and descending order.

The Sort Column indicators are by default indicated by encapsulating the column header text with < and > You can use any other indicators you want, but the classes only support one character indicators for prefix and postfix. This can be changed in code by using the SetOrderSign and SetSortSign methods and then call the UpdateHeaders method to refresh the header displays. The UpdateHeaders method should be called if the headers need to be changed for any reason. It will update the class properties with the current properties of the hederss. That way when the sort order changes for example, the column headers are still correct. This also provides full support for runtime translations of the browses.



ClarioNet Alertkey

This option is only available if ClarioNet is being used. By default ClarioNet uses the Left Mouse button which means it can not be used to change the sort order in the browse if it is running under ClarioNet. Just select a different mouse combination here and you can run the application under ClarioNet. The default key combination is AltMouseLeft.

FILTER

On this tab you can specify filters for the entire browse or for individual columns.

Global filter/range limit

The Global filter is a filter that applies to the whole browse, regardless of what column filters may be applied. For example if you need to filter a browse based on a parent value, a Global filter is what you need to use as it will filter the browse on that value only and never show anything else. If you need to further limit certain columns to other values, you need to add column filters as well.

You need to check the Filter Scope setting to make sure that the startup scope is set correctly.

By default the filter is quoted by the templates. This can make it difficult to put in clarion variables and have them resolve properly. In version 6.0 you can now start the filter with an exclamation mark and the filter will be generated exactly like you put it in, for example:

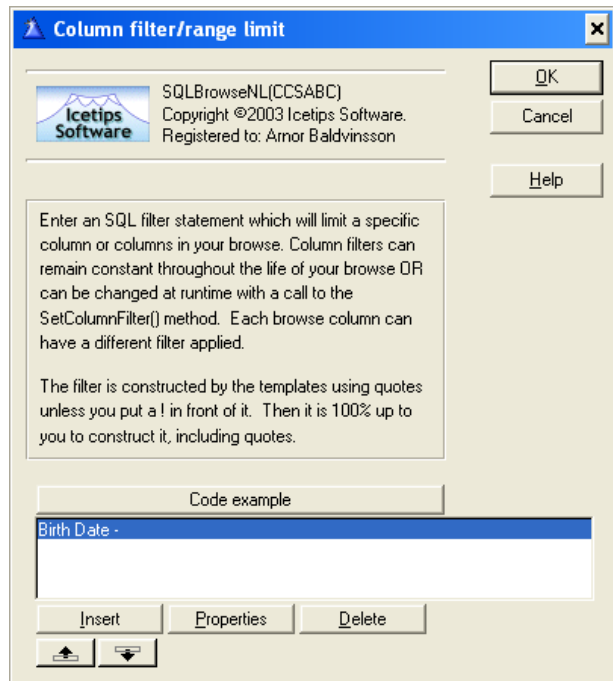
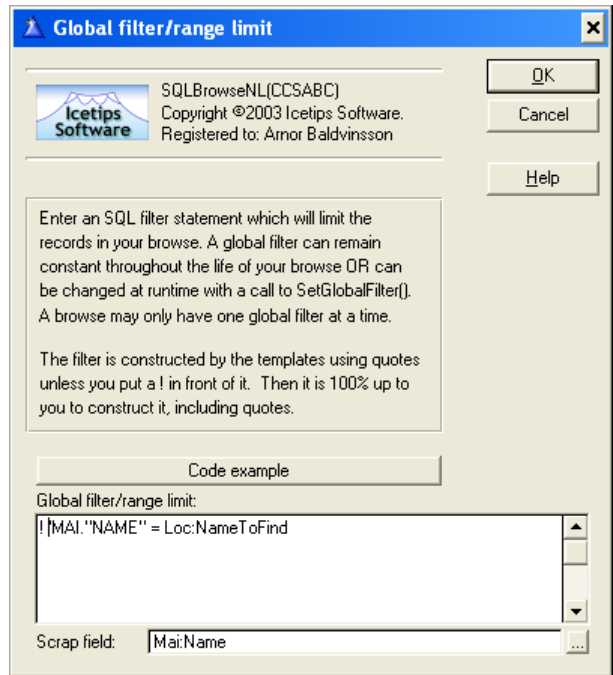
```
!'MAI."CONTACT" = ' & Loc:ContactID
```

This will then resolve to the appropriate filter and it is completely up to you to make sure that this generates the correct filter!

You can change the Global filter at any time during the lifetime of the browse by using the SetGlobalFilter method or the ForceGlobalFilter which forces an immediate refresh of the browse. You can also use the ClearGlobalFilter method to clear the filter.

Column filter/range limit

The Column filter applies to the browse column that is the currently active sort



NEW

column. For example if you only want to show records for the state of Texas, when the state column is the sort column, you could use a column filter for the state column. When the browse is sorted on the State column, it will then only show the records for the state of Texas.

You need to check the Filter Scope setting to make sure that the startup scope is set correctly.

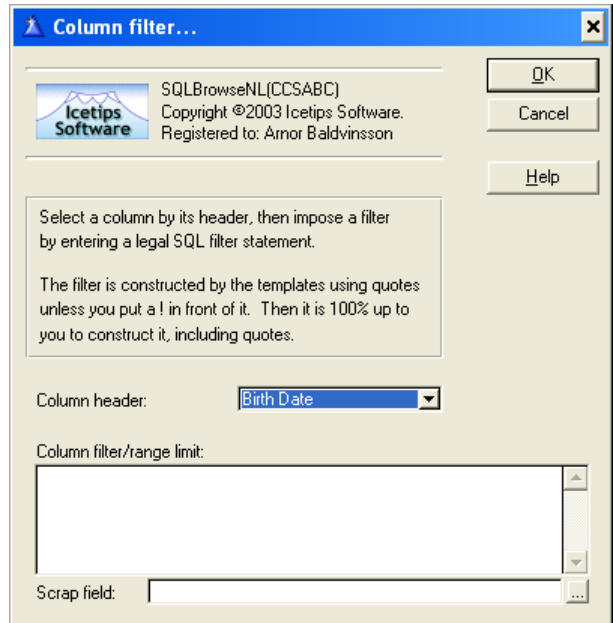
By default the filter is quoted by the templates. This can make it difficult to put in clarion variables and have them resolve properly. In version 6.0 you can now start the filter with an exclamation mark and the filter will be generated exactly like you put it in, for example:

NEW

```
!'MAI."CONTACT" = ' & Loc:ContactID
```

This will then resolve to the appropriate filter and it is completely up to the developer to make sure that this generates the correct filter!

You can change the Column filter at any time during the lifetime of the browse by using the SetColumnFilter method or the ForceColumnFilter which forces an immediate refresh of the browse.



Filter scope setting

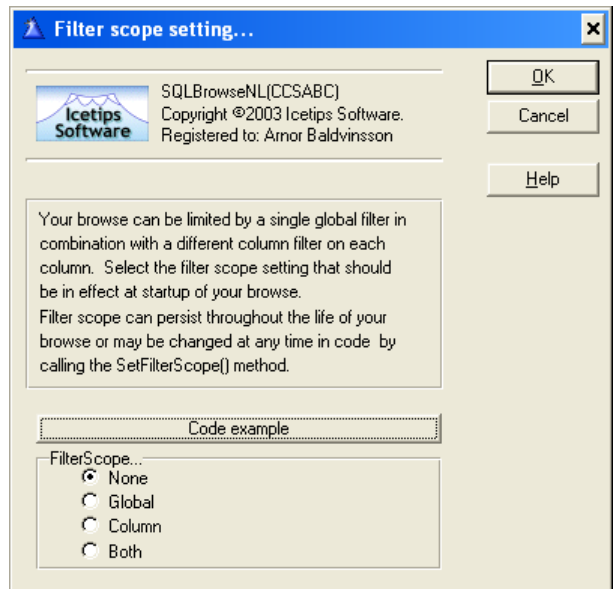
The Filter Scope setting sets the initial filter scope for the browse, used when the browse starts up. If you only have a global filter, check the Global option in the Filter Scope settings. If you only have Column filter, check the Column option in the Filter Scope settings. If you have both global and column filters, check the Both option in the Filter Scope settings.

You can change the filter scope at any time using the SetFilterScope method. It can take one of 4 possible values as they are defined in the CcsSql1.inc file:

```
HPROP:FilNone           EQUATE (1)
HPROP:FilGlobal        EQUATE (2)
HPROP:FilColumn        EQUATE (3)
HPROP:FilBoth          EQUATE (4)
```

To change the filter scope setting, call the SetFilterScope with one of these equates. For example to set both Global and Column filters, you can use:

```
SQL1.SetFilterScope (HPROP:FilBoth)
```



To set only a Global filter, you can use:

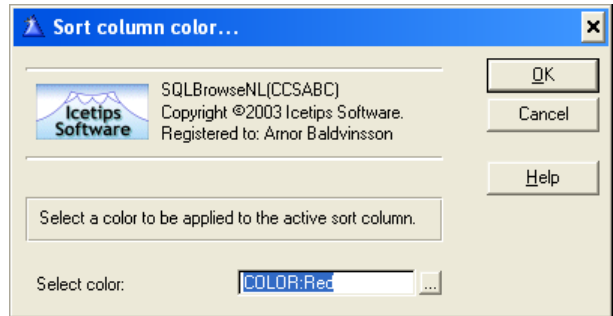
```
SQL1.SetFilterScope(HPROP: FilGlobal)
```

SORT

This tab has settings that apply to how the sorting options are handled in the browse.

Sort column color

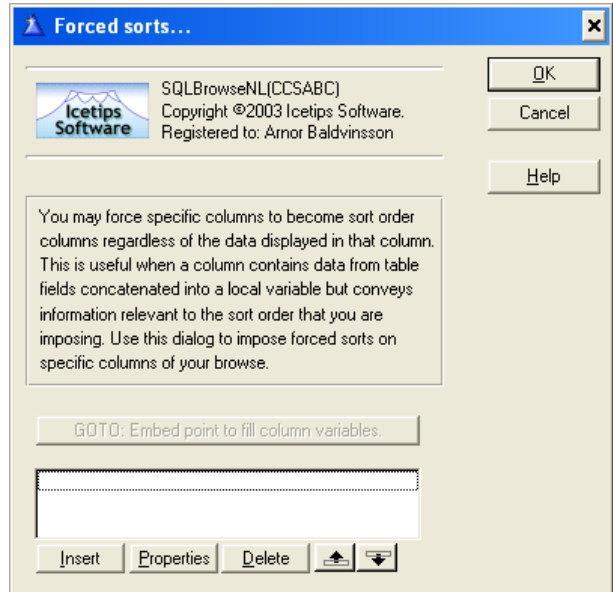
The Sort column color is used to indicate what column is the active sort column in the browse. When a column becomes the active sort column, the text color in that column is changed to the color specified here. By default it is set to COLOR:Red, but you can set it to anything you want. Please note that in version 6.000 the Sort column color is overridden by the Greenbar settings, but overrides the color settings. So make sure that you test this with the Greenbar setting if you set the foreground color using the Greenbar. Please refer to page 16 for more information on the Greenbar settings.



NEW

Forced sorts

The Forced sorts allow you to use a particular key or field as a sort field for another column if the data in that column can not be used to sort the browse, for example if the data is from local variables, or if the data doesn't lend itself to intelligent sorting for whatever reason.



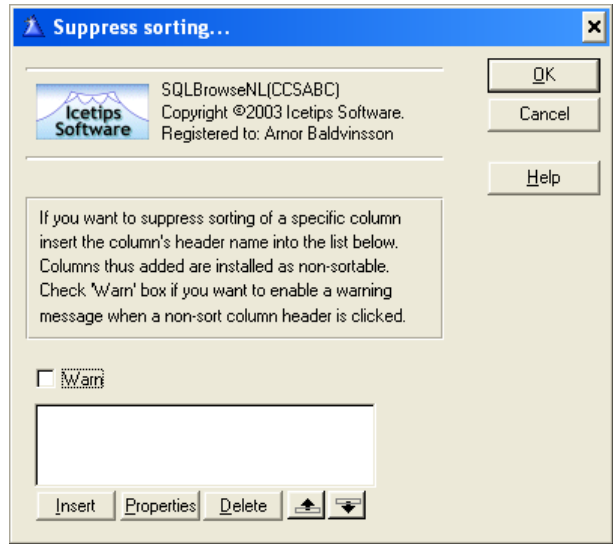
NEW

In version 6.000 the need to do this for non-table fields no longer exists as we have implemented an automatic way to deal with this for local variables. Please refer to the section about the Variables on the Data Access tab on page 26 for more information about how this new feature works. In previous versions you would need to add local variables to the Forced sorts.

You can also use this to simply force a column not to sort on it's own data but on something else. An example could be if you have a FirstName, LastName and FullName fields in a database. You may not want the FullName field to sort on it's own contents, but on the LastName. In that case you can use the Forced sort to force the FullName column to sort on the LastName.

Suppress sorting

By adding columns to the Suppress sorting list, you are disabling the header click sorting on that column. If the user clicks on the column heading, nothing happens except if the Warn is checked, a message will be shown to warn the user that the column can not be sorted. This can come in handy on columns that can not be sorted, for example columns with local variables representing calculated or concatenated data. Use the Insert button to add columns to suppress sorting on.

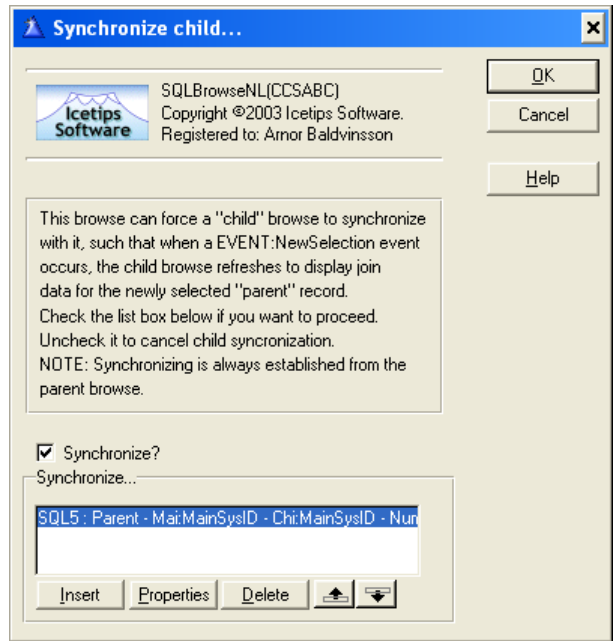


JOIN

This tab has options that affect how listboxes are joined and how the Where clause is constructed.

Synchronize child

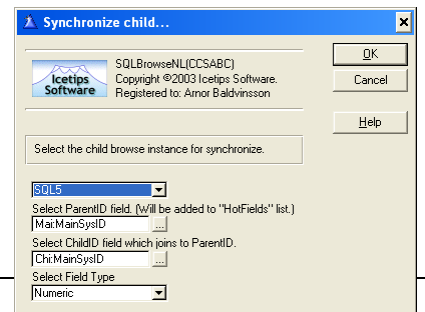
This is used on a parent browse to synchronize a child browse so it refreshes correctly when the parent browse is scrolled. This is the same as setting range limits on ABC browses to range on a single value from the parent browse. In ABC this is always done on the child browse, but in the SQL it is always done on the parent browse. To start with, check the Synchronize checkbox to turn it on. Then click on the Insert button to add a new synchronization. Select the child browse you want to synchronize from the dropdown of available SQL browses on the window. Then select the parent and child fields and select if these fields are numeric or alphanumeric. Usually these would be system id fields. You can have multiple browses synchronized by adding more browses into the listbox at the bottom of the window.



NEW

Synchronize Multi child

In some cases it is necessary to work with primary keys with more than one component. This is most important when dealing with some sort of replication on the back end where publications and subscriptions are limited to a certain ID field. In this case you need multiple fields to synchronize the browses if the parent primary key contains more than one field. The Synchronize Multi child is a new option in version 6.000



and allows you to do exactly that. However, in this initial form it is limited to synchronize only ONE child browse! We will add the option to synchronize multiple child browses with this option in our next release.

Please refer to the section on code samples starting on page 39 for more examples on how to synchronize child browses with single and multi component primary keys.

WHERE clause

This options allows you to add to the built Where clause and allows you to create complex Where clauses. This works together with Global filters and Column filters as well as the Locator filters, i.e. the where clause you add here is in addition to all the other filters that also affects the Where clause. This gives you very powerful options to customize the browses and add customized filters. You can also do this in code by using the SetWhereClause method.

To create a Where clause, click on the "Expression Builder..." button and then click on the Insert button to start adding new fields. Specify the left and right hand side of the assignment and the operator.

The Resolve option makes it possible for you to put variables as the right side of the assignment and it will be generated correctly into the resulting string. Without the Resolve option, the right field is generated as if it was an SQL field i.e. FIL."Fieldname" but with Resolve it would be generated as FIL:FieldName. Examples:

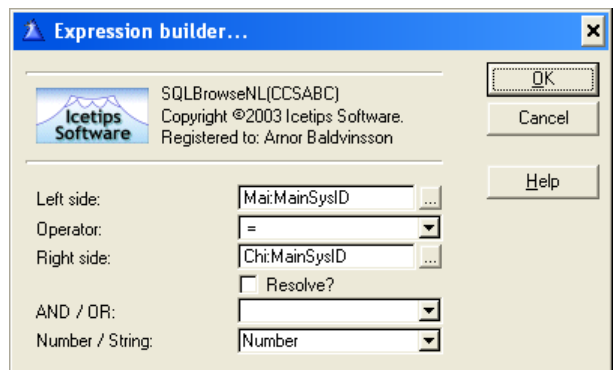
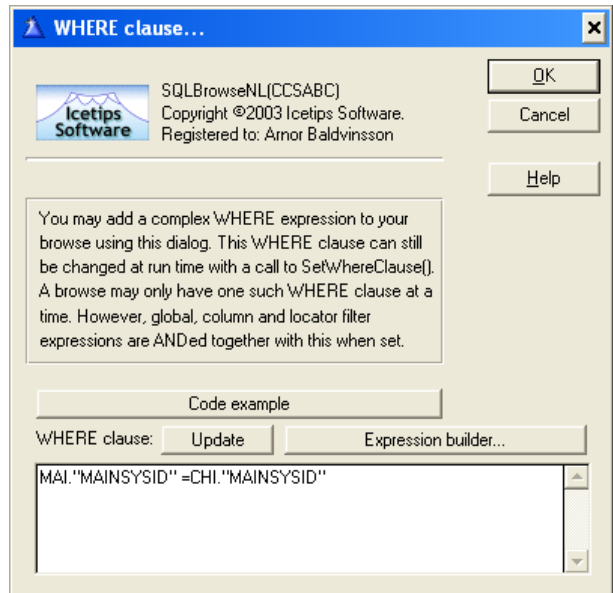
With Resolve:

```
SELF.SetWhereClause('MAI."NAME" ='' & MAF:NAME & ''')
```

Without Resolve:

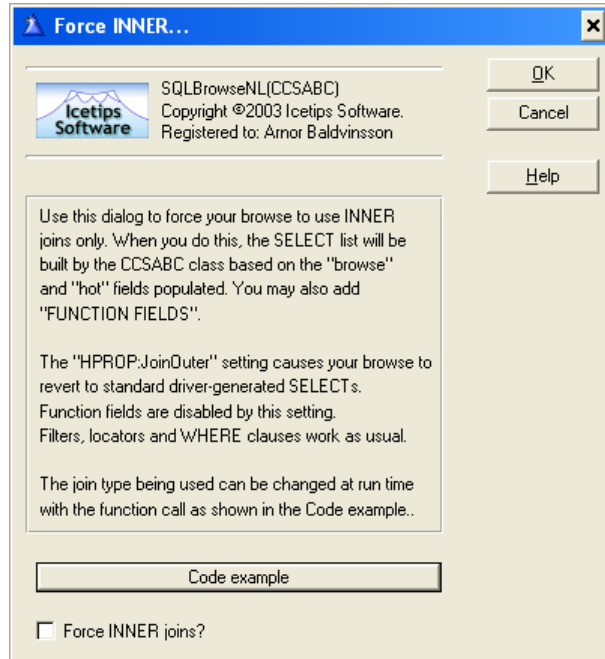
```
SELF.SetWhereClause('MAI."NAME" ='MAF."NAME"''')
```

Select the AND/OR option if you want to add another part to the Where clause. If this is the only part or the last one, leave it empty. Finally select the datatype, string or numeric. When you are back in the main window, click on the "Update" button to bring in the expression that you have built. You can enter the expression directly on the main window, but using the Expression Builder reducing spelling and typing errors.



Force INNER

This is another very powerful option. With the Force INNER checked, the SQL template generate the entire Select statement and use PROP:SQL to retrieve the data from the backend database. With the Forced INNER you can also use function fields to retrieve data such as aggregate functions like SUM() or COUNT(). Please refer to the Function Fields options on page 24 for more information about using functions. To turn it on, simply check the Force INNER or you can change this at runtime using the SetJoinType method and set it to either HPROP:JoinInner or HPROP:JoinOuter.



DATA ACCESS

This button holds various very powerful settings and options among other things hot fields, stored procedures, reset fields and (local) variables.

Hot fields

This is identical to hot fields in ABC/Clarion browses, i.e. fields that are not part of the listbox, but are retrieved from the database and are therefor part of the Select statement. If you need fields for calculation into local fields, system ID fields for linking child browses or whatever, this is the place to add them. Just click on the Insert button to add hot fields.

NEW

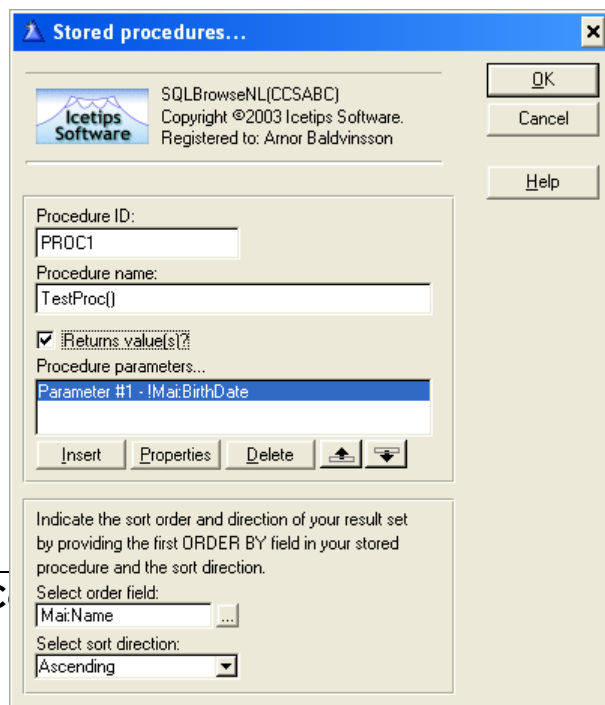
Fill on demand

This is replaced with ActiveInvisible in version 6.000 but left in here for backward compatibility. Please refer to the ActiveInvisible section on page 26 for further information.

Stored procedures

You can add information about stored procedures that you have in your database by adding them here. You can add multiple stored procedures to your browse and use them with the SetViewProcedure or ForceViewProcedure methods.

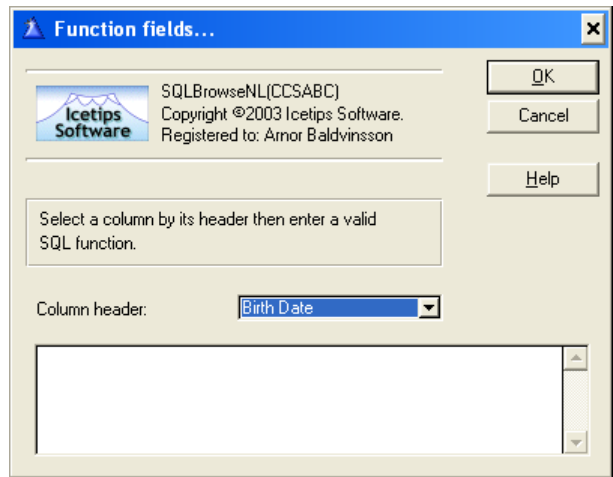
A procedure ID is generated for you automatically, but you need to specify the procedure name and any parameters that it needs. You can also specify the first ORDER BY field in the stored procedure and if the sort order is ascending or descending.



Function fields

Function fields can be used to put the results of SQL functions into specified columns in the browse. This option only works if the Force INNER is active as then the SQL templates build up the entire SQL statement. Please refer to page 23 for more information about Force INNER.

To add a function field, click on the "Insert" button and select the column you want to use. Then type in the SQL function you want to call. Please note that the database field for the column that receives the results must have the same datatype as the function returns. For example if you call the Count() function, the field data type must be an integer data type.



Extended SQL

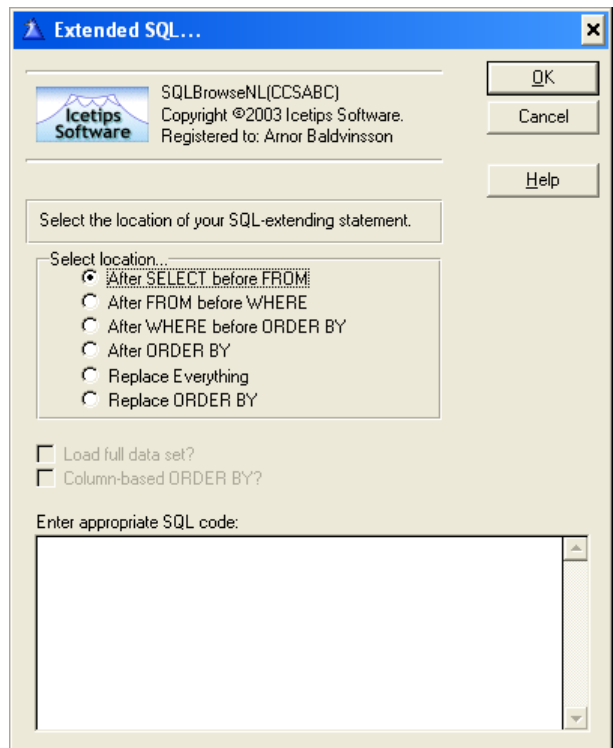
This is an extremely powerful option that let's you append your own SQL statements into the SQL generated by the SQL classes. You have 6 options for what you can change.

After SELECT before FROM

This appends the SQL you enter to the SELECT statement. For example if you add , MYF."Field3" as extened SQL and the generated SELECT statement looks like **Select MYF."Field1", MYF."Field2"** the resulting SELECT will be: **Select MYF."Field1", MYF."Field2", MYF."Field3"** This way you can add fields into the Select statement.

After FROM before WHERE

This appends the SQL you enter to the FROM statement. This is handy to use to add INNER joins without specifying the table in the Clarion view. Example, if the Clarion generated Select statement is **Select Namesysid, Name, Address, City, State From Names** but you want it to be **Select Namesysid, Name, Address, City, State From Names ,Orders where Namesysid = Orders.CustSysid** then you add **ORDERS** in "After FROM before WHERE" and add **Nam.Namesysid = Orders.CustSysid** in "After WHERE before ORDER BY"



After WHERE before ORDER BY

This appends the SQL you enter to the WHERE statement. You can use this to add fields to the WHERE clause if needed.

After ORDER BY

This generates the SQL statement you add into the ORDER BY statement. These fields are listed prior to other fields in ORDER BY.

Replace Everything

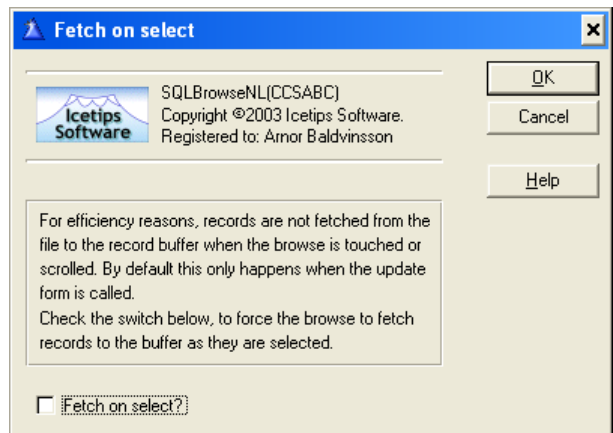
This replaces the generated SQL completely with what you enter, so you can build up your SQL statement completely independent of what the templates and classes would generate. This is only for developers who have a good understanding of SQL and how it relates to Clarion views and browse queues.

Replace ORDER BY

This replaces the entire ORDER BY clause.

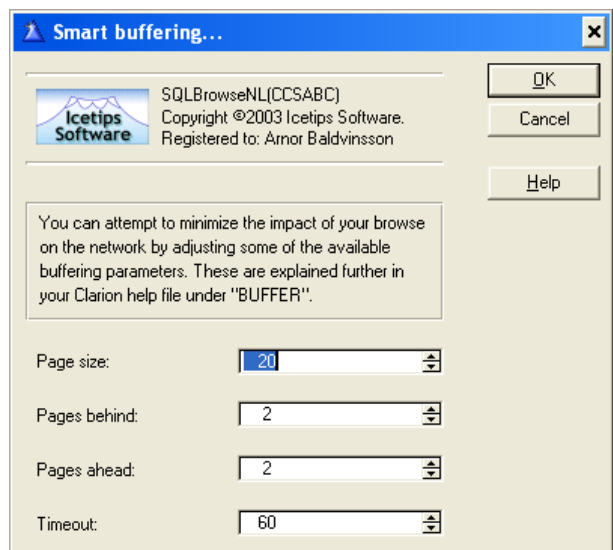
Fetch on select

This is used to force a select of the current row from the database every time a row is selected in the browse. Normally this is not done to reduce network traffic. By using this, the browse will force an update of the currently selected record from the database each time you select a record by scrolling or clicking with the mouse on the listbox.



Smart buffering

The SQL templates use the Clarion Buffer statement to set up buffers for the browse. We discovered too late that these settings have not been implemented in the SQL templates, but we intend to implement them if necessary in the next release and are therefor documenting them here. These settings are:



Page size

An integer constant or variable which specifies the number of records in a single "page" of records. The default value is 20.

Pages behind

An integer constant or variable which specifies the number of "pages" of records to store after they've been read. The default value is 2.

Pages ahead

An integer, constant or variable which specifies the number of additional "pages" of records to read ahead of the currently displayed page. The default value is 2.

NEW

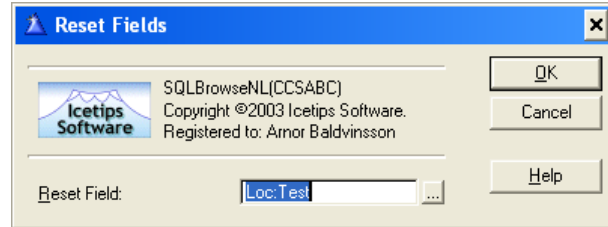
Timeout

An integer constant or variable which specifies the number of seconds the buffered records are considered not to be obsolete in a network environment. The default value is 60 seconds.

NEW

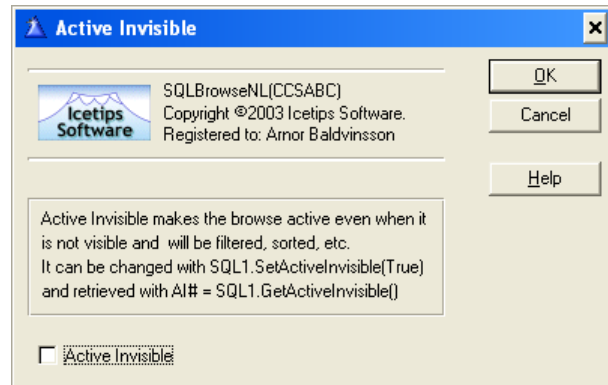
Reset Fields

The reset fields are a new feature in version 6.000. The reset fields take advantage of the ABC Fieldspairs class and the WindowComponent interface implemented in version 6.0. You can add any variable into the reset fields and if the value of those variables changes, the SQL browse will be notified and refreshed as needed. You can use local, module or global data or you can use database fields.



Active Invisible

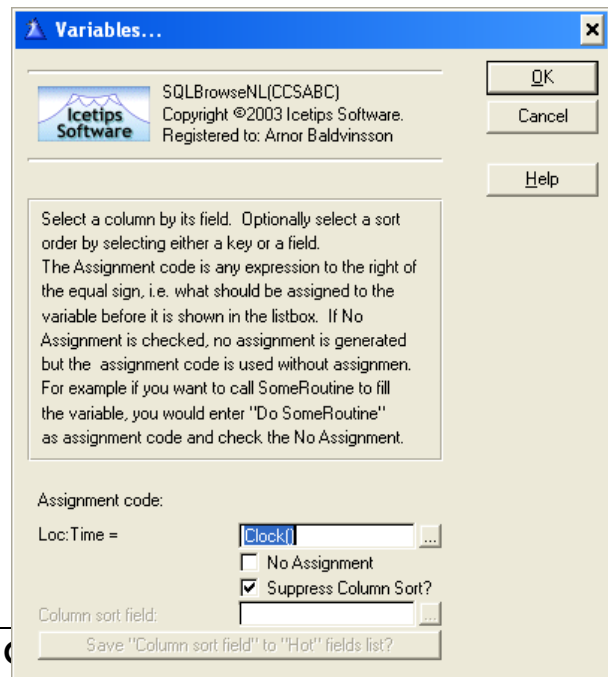
Active Invisible is a new feature in version 6.000. It overrides the old "Fill on Demand" setting. With Active Invisible set the listbox is always active and will refresh and refill as needed even if it is not visible. If Active Invisible is unchecked, the listbox is only refreshed and refilled as needed when the browse control is actually visible. Unchecking Active Invisible makes complex browse windows with many SQL browses load faster as they do not need to refresh the invisible browses on startup. The default for this setting is ON to maintain compatibility with older browses.



NEW

Variables

This is a new option in version 6.000. If there are non-database fields in the browse box, they are automatically added to the list of variables. A generic sort column is defined and it is the first column of the primary file that is available in the listbox. Sorting on variable columns is turned off by default. By default the variable is simply assigned to itself, and should thus not break any existing code that uses Forced Sorts. For



example if you have a variable called Loc:Total in your browse, version 6.000 will simply generate this into the FillQueueField method code:

```
Loc:Total = Loc:Total
xValue = Loc:Total
```

The Generic Column Sort Field is used to define which columns should trigger updating what fields. This is by default the first browse column from the primary file. Each variable can have a sortfield specified, but column sort is turned off by default for the local fields. None of this should affect existing code for local variables. If you find cases where this causes problem, please report it to us immediately via email to support@icetips.com and we will fix it right away. Our tests have not turned up any migration problem in this area, but this is a new feature so it is quite possible that it may have problems somewhere.

Assigning values to the local variables can be done either with a direct assignment by selecting something into the assignment field. You can select multiple variables and enter other information there as expressions. You can also check the "No Assignment" option if you want to call a routine to assign a value to the variable. In that case no assignment is generated, just the code you enter into the assignment field. For example if you check "No Assignment" and enter "Do CalculateRoutine" into the assignment, this will be generated as:

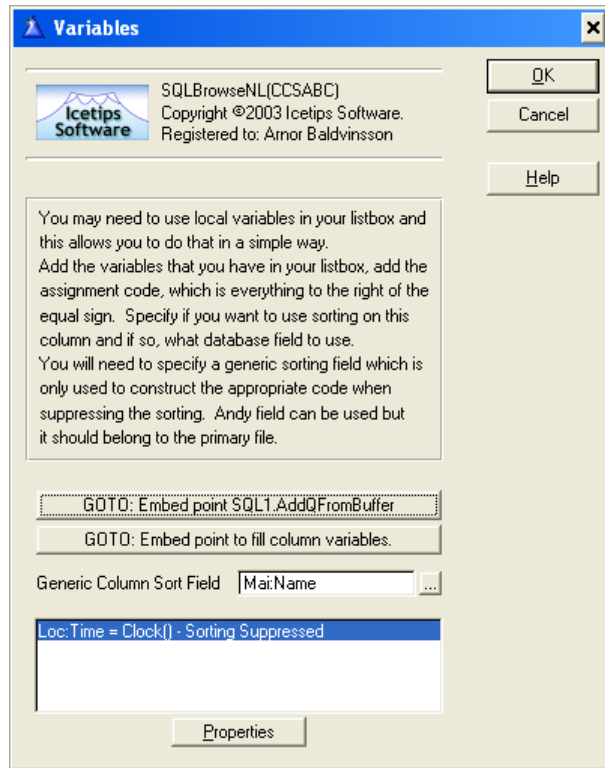
```
Do CalculateRoutine
```

If you on the other hand uncheck the "No Assignment", that same call would be generated as:

```
Loc:Total = Do CalculateRoutine
```

Which would cause compiler error.

You can use the two GOTO buttons to open up the two embed points that can be used to manually fill the variables. The first one goes into AddQFromBuffer method and the other goes into the FillQueueField method.



NEW

Force Use of the - primary- file and related files

This option can be used with the LazyOpen option which can be set at global level in the ABC templates - "Defer opening files until accessed" which you can find on the "File Control" tab of the global properties window. By forcing the opening of the primary file, problems with the LazyOpen are solved. This option is new in version 6.000. Default value is true. This results in code being generated in the WindowManager.Init method:

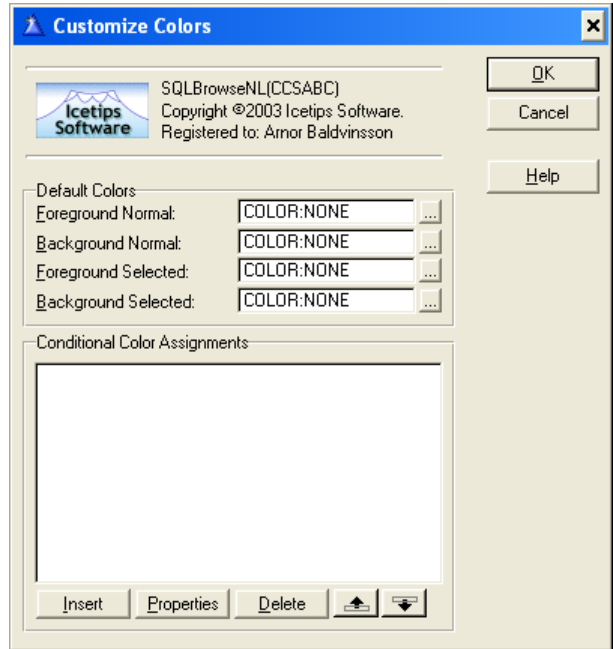
```
Access:PrimaryFile.UseFile
```

Access:SecondaryFile.UseFile
 Access:SecondaryFile2.UseFile
 Etc.

NEW

COLORS

This option is for all its intents and purposes completely identical to the ABC and Clarion browse colors options. It will list the columns in the browse that have the color setting turned on in the Listbox Formatter and you can select what colors to use here. Note that the Greenbar effect overrides the Colors options in version 6.000. Please refer to the Greenbar on page 16 for more information about the Greenbar effect. Conditional colors can be specified in exactly the same way as in the standard browse templates. This option is new in version 6.000.



NEW

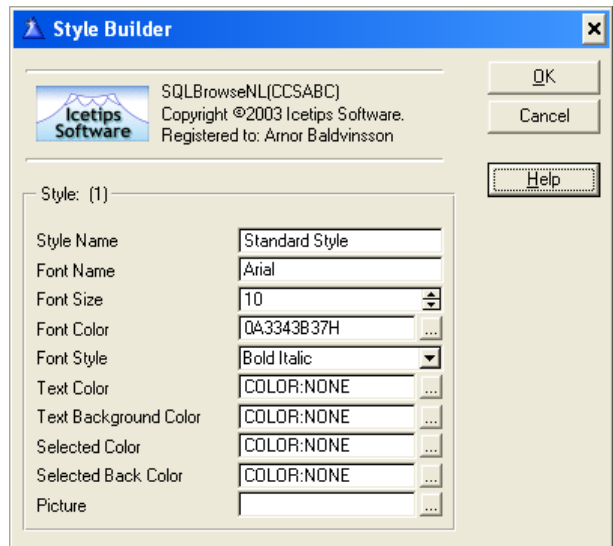
ICONS

This option, like the colors is identical to the icons in ABC and Clarion browse templates. It will list the columns in the browse that have the icons turned on in the Listbox Formatter. Conditional icons can be specified in exactly the same way as in the standard browse templates.

NEW

STYLES

Styles are a very powerful tool to format browses and listboxes to your liking. You can change fonts, colors and data picture by specifying a style. Styles are new in version 6.000 and we have provided a simple Style Builder to make life easier for you. Click on the "Style Builder" button and then click on the Insert button. Give the style a name and specify the settings you want to set. You can set any or all of the options available, but only the ones that are set will be generated. Once you have created the styles you want, you can apply them to the columns in the browse that have styles enabled, simply by selecting the style name from a dropdown. You can set conditional styles in the same easy manner by specifying the condition and pick the style to use. This way you can easily show specific data using a specific style. For example subscription that is run out or overdue you could show in red with a bold font. Show payments in green.

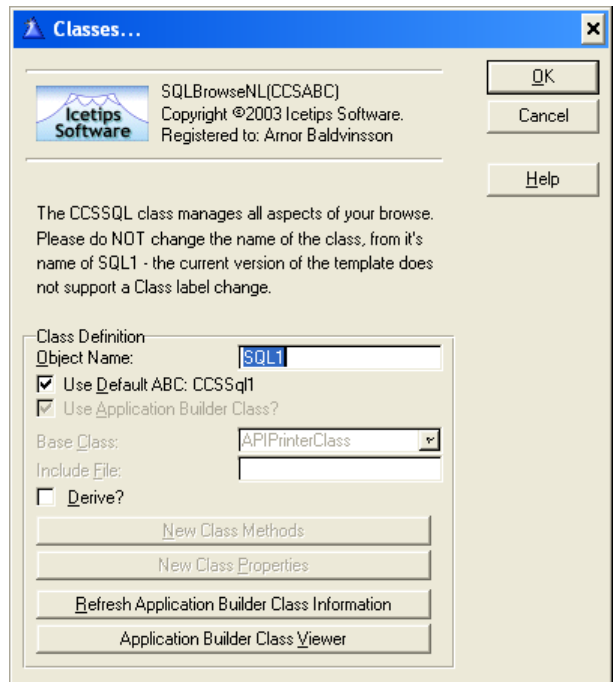


NEW TOOLTIPS

The tooltips are a new feature in version 6.000. Too late to be fixed for this release, we discovered that they do not seem to work, neither on SQL browse or plain ABC browses. We will investigate this for the next release.

CLASSES

The classes tab only have the "Classes" button which takes you to the standard ABC classes window. That window allows you to change the object name. **Please do NOT change the object name** in the current version (6.000) It will break the generated code as it expects the name SQL and the instance number of the active template. You can however use this option to derive and add new class methods and properties if you need to.



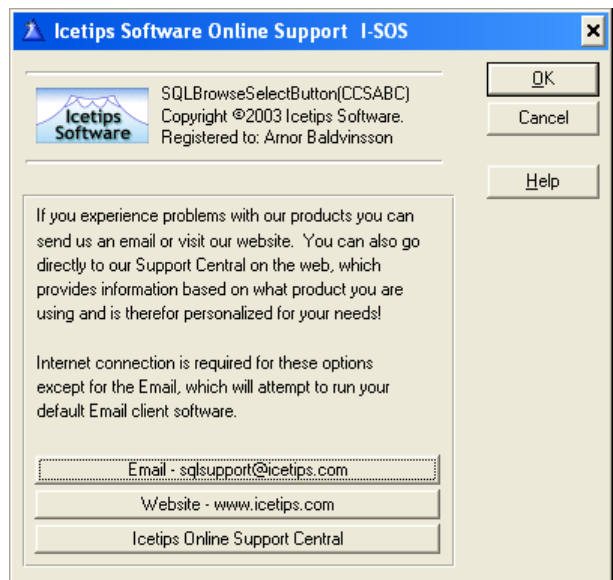
NEW Icetips Software Online Support I-SOS

This window has 3 buttons that give you direct access to online support. The SQL templates version 6.000 is the first of our products that get's this new support options which we will add and include in all our products as we release new versions and builds in 2003.

The first button starts up your default email client and props it with the appropriate email address to send your support request to along with some other information. Please do not delete any of the information that is automatically populated as it may delay support!

The second button will take you to our website at <http://www.icetips.com> where you can check out our online resources for SQL and also other of our products.

The third button will take you directly to our support center which we are developing. To begin with you will only get a page where you can type in a message to us, but in the near future we will introduce FAQs and other online resources for our customers.



Icetips SQL Browse Locator and CCS SQL Browse Locator

The Icetips SQL template contain two slightly different locator controls. One is identical to the original locator template and is provided for backward compatibility and because it provides a slightly different functionality. The old locator which is called "CCS SQL Browse Locator" makes it possible to use one locator control for all browses on the window. This is referred to as a shared locator. In some cases this may not be intuitive for the end user and that is why we also provide the new "Icetips SQL Browse Locator" which works with a specific browse only and can exist in multiple instances on the same window.

Shared Locator

This only applies to the "CCS SQL Browse Locator". By checking this option you enable multiple browse boxes to use the same locator.

Progressive Locator

A progressive locator is a drill down locator where each search is ANDed with the previous locator value. This allows you for example to locate TX for Texas in a statefield and then locate San Antonio in a city column to retrieve all records with both TX and San Antonio. The locator continues to be built up until the user hits the Clear button at which point it is cleared completely. You can change the locator at runtime by using the SetLocatorType method and use either HPROP:LocSimple or HPROP:LocProgressive, defined in the ccsql1.inc:

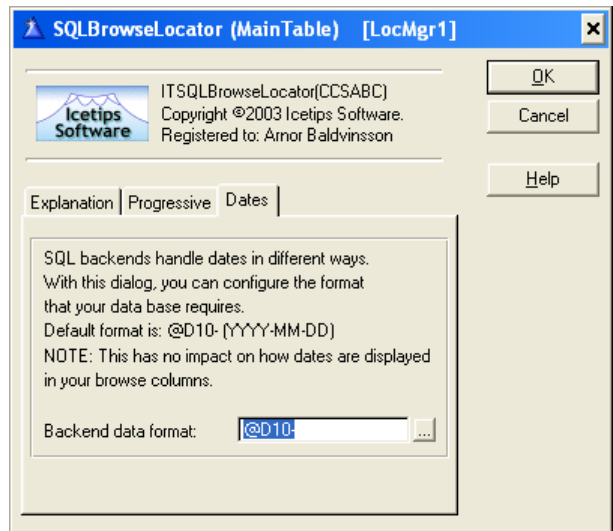
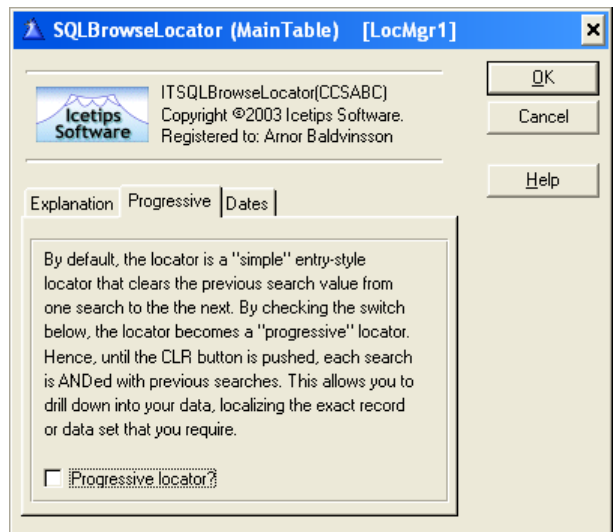
```
HPROP:LocSimple EQUATE(5)
HPROP:LocProgressive EQUATE(6)
```

Example, to set the Locator to Progressive Locator in code use the following:

```
SQL1.SetLocatorType |
(HPROP:LocProgressive)
```

Dates in Locators

The way the backend get's date values is very important. By default the date value for the locator is set to @D10- You may need to change this if you experience problems with locators on date columns. You can set the locator date picture at runtime by setting the BackEndDateFormat property. Example, if you want to conditionally change the format to @D12-



you could use this:

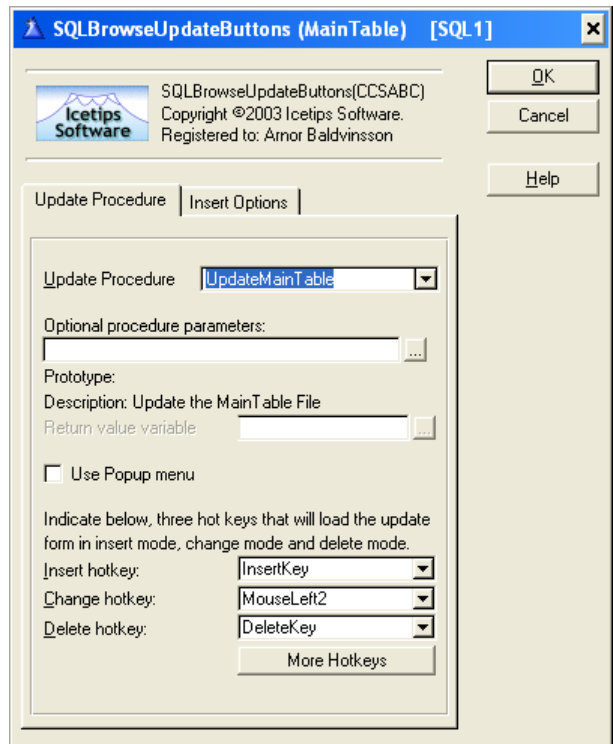
```
If BackendWantsD12
    SQL1.BackEndDateFormat = '@D12-'
Else
    SQL1.BackEndDateFormat = '@D10-'
End
```

Icetips SQL Update Buttons

Update Procedure

SELECTING UPDATE PROCEDURE

Here you select the update procedure to use for the browse. Select the form from the dropdown, which will show you all procedures in the application. If the form takes parameters, you can add the variables to pass to it. To assist you with that the template will display the prototype of the form and return type as well as the description of the form procedure. If the form returns a value, the Return Value variable will be enabled so you can specify a variable to receive the return value from the form.



NEW

POPUP MENU

This is a new option in version 6.000. Now you can implement the popup menu in the same way as in the ABC browse. The popup class will automatically take care of handling the popup menu. The popup class is set up to mimic the update buttons. You can access the Popup menu class with SQLx.Popup, where x is the instance number of the SQL Object. For example if you want to add your own button to the popup menu, you can use:

```
SQL1.Popup.AddItem('-')           ! Adds a separator
SQL1.Popup.AddItemMimic('Button5', |
                               ?Button5)   ! Add a button mimic
                                           ! using button text as menu text
SQL1.Popup.AddItemMimic('Export', |
                          ?ExportButton, |
                          'Export data')    ! Add a button mimic
                                           ! providing text for menu text
```

NEW

HOTKEYS

This section specifies the keys to alert on the browse to perform the insert, change and delete actions. By default these are set to InsertKey, MouseLeft2 and DeleteKey. In most cases this is enough, but in many cases you may want to add a hotkey to perform some of the tasks. In version 6.000 you can add as many hotkeys as you want to use with the update buttons. Each key requires an action to be selected so the browse knows what to do when the user hits the key. You can add hotkeys directly in code by using the RegisterKey method. It takes the keycode and action as parameter. The actions are the standard template actions of InsertRecord, ChangeRecord, DeleteRecord or SelectRecord. To add the enter key as an example, you could do it like this:

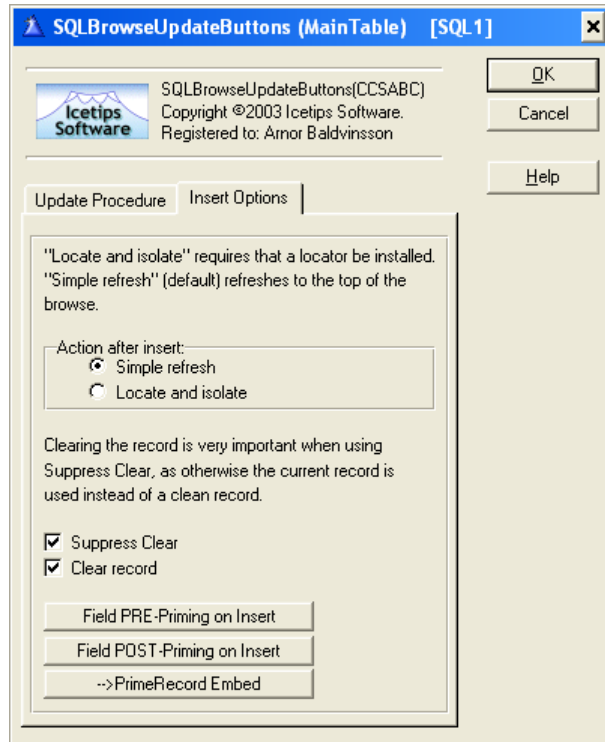
```
SQL1.RegisterKey(EnterKey, ChangeRecord)
```


Insert Options

This tab has options that relate to what happens when a record is inserted.

LOCATE AND ISOLATE

This option makes the browse locate the newly inserted record and isolate it. This means that the new record is the only record visible in the browse box. In future release we intend to make the templates automatically reset to where the new record is and refresh so that it is selected. This, however, requires major changes to the classes and we decided not to do it for this release.



NEW

SUPPRESS CLEAR

This is a new option in 6.000. When checked the record is not cleared when the record is primed in the FileManager PrimeRecord method. This means that fields can be primed prior to calling the PrimeRecord method of the ABC FileManager. This allows you to prime fields that are not allowed to be NULLs when the record is added. If you are using Clarion's autonumber on files, the record is added when it is primed. If the record is cleared before it is added, it may contain invalid fields, for example where Referential Integrity is enforced. Using this option with the Field Priming options, you can have the templates generate code as the following code, from a working application:

```

Clear(IOI:Record)
Loc:NewIOIID          = GetAutoIncNumber (Loc:BrokerID, |
                                          EQU:AutoNum:IOI, |
                                          Glo:IsOfficeProgram)

IOI:BrokerID         = Loc:BrokerID
IOI:ContactID       = CON:ContactID
IOI:AccountID       = ACC:AccountID
IOI:TransactionID   = Loc:NewIOIID
IOI:CompanyID       = BRO:CompanyID
IOI:OfficeID        = BRO:OfficeID
IOI:ProgramID       = Glo:IsOfficeProgram + 1
SetNull(IOI:CusipSysID)
SuppressClear = True
! Parent Call
ReturnValue = PARENT.PrimeRecord(SuppressClear)
    
```

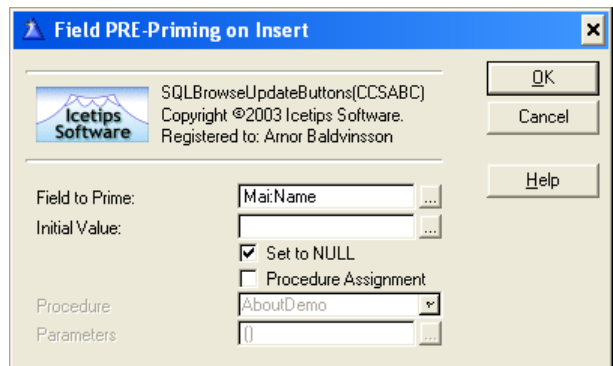
This uses Pre-Priming, as well as SuppressClear and the option to Clear the record before it is primed. It is very important that the Clear is used when using SuppressClear as otherwise the active record is used to prime the new record! SuppressClear makes it possible to set up complex priming both before and after the PrimeRecord is called.

NEW

FIELD PRIMING

Field priming can occur in two places, before and after the PrimeRecord method is called. The PrimeRecord method calls the FileManager PrimeRecord, which will attempt to add the record to the database. By using the SuppressClear, Clear, pre- and post-priming, it is now possible to prime the record with what you need to make things work smoothly.

Example: You have a field that is set to CAN NOT BE NULL constraints in the database. Now you can simply populate it with some valid value before the record is primed, and then use SETNULL() on it after it is primed to enforce the constraint when the form is accepted. It will not allow the user to close unless he or she puts something into that field!

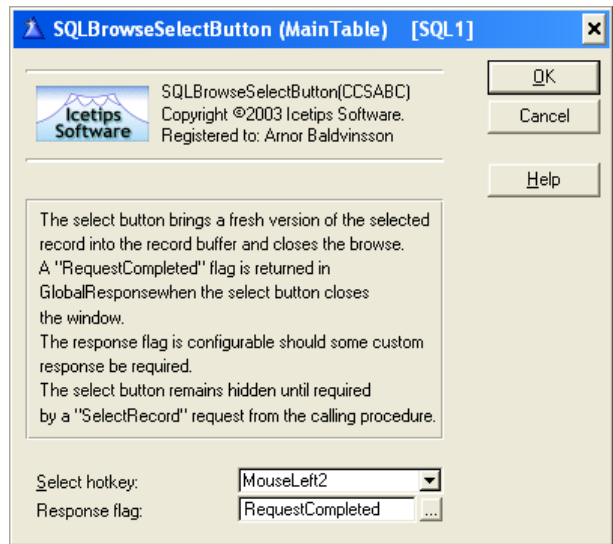


Icetips SQL Select Button

The Select button control template simply puts a button on the window that allows the user to select a record and return it's buffer to the calling procedure. This button is hidden unless the browse is called with SelectRecord, example:

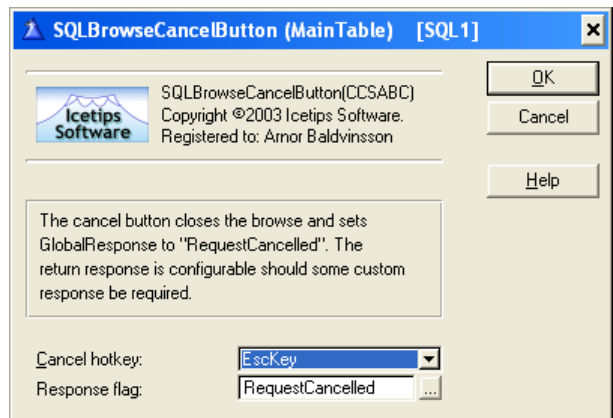
```
GlobalRequest = SelectRecord
SQLBrowse ! Call the browse
If GlobalResponse=RequestCompleted
    ! A record was selected
Else
    ! Cancel on browse
End
```

This works identical to the Select button control template in the ABC templates.



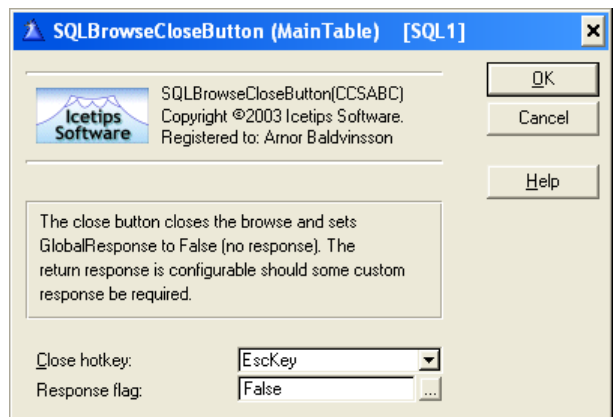
Icetips SQL Cancel Button

This control template adds a Cancel button to the window. The template generates a call to the InitCancel method of the CCSButtons class. This simply makes the classes aware of the Cancel button. We have not experienced any problems in using the standard ABC Cancel button with the SQL templates.



Icetips SQL Close Button

This control template adds a Close button to the window. The template generates a call to the InitClose method of the CCSButtons class. This simply makes the classes aware of the Close button. We have not experienced any problems in using the standard ABC Close button with the SQL templates.



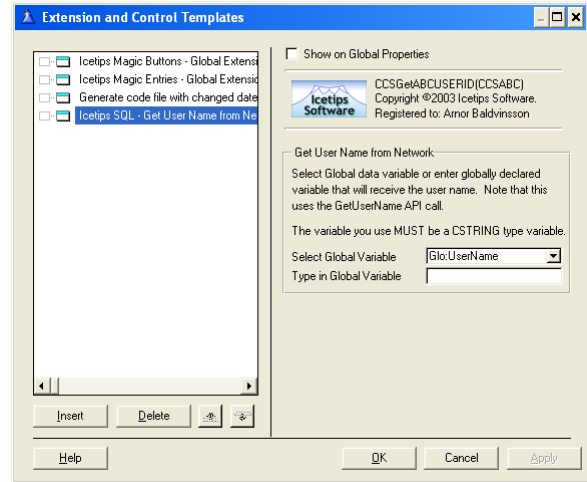
Drop list of Sort Orders

No documentation available at this time.

Global Extension Templates

Get User Name from Network

This template uses the GetUserName API call to retrieve the user name of the currently logged in user from the computer. It requires a global variable to be picked which will receive the username. This is performed in global code before the main procedure is run, so the username is available immediately upon entering the main procedure. Note that the variable used to contain the user name MUST be a CString. You can select a variable from the global data or you can enter a variable that is declared in code and is not available in the global data dropdown list.



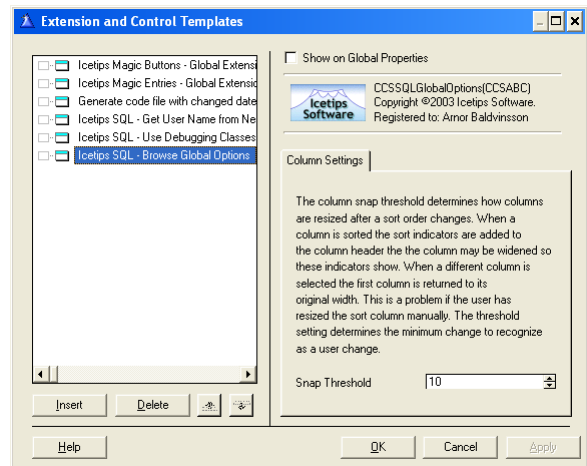
Use Debugging Classes

The debugging classes are no longer included in the product and this template is only provided for backward compatibility.

The Icetips SQL classes include a method called wDebug that can be used to pass strings to OutputDebugString API call. The output can then be viewed with debugging tools such as the freeware DebugView from www.systeminternals.com. Please refer to the wDebug method of the in the Class Reference on page 84 for more information.

Global Options

This template has only one setting. It determines what is the minimum change in column width that is regarded as user change. When the sort order moves from one column to the other, the order signs and the header determine how wide the column needs to be to be able to show both the header and the order signs. The font setting for the browse or window also affect the width. To change the threshold that determines the minimum change considered to be user change and means that the column width will NOT be reset by the SQL templates, select the appropriate number to use. The default is 10, which means that if the user changes the size of the column by 11 dialog units, the templates will not change it when the header changes. If you reduce the number it means that the user can make smaller adjustments to the columns without the SQL templates changing the column widths.



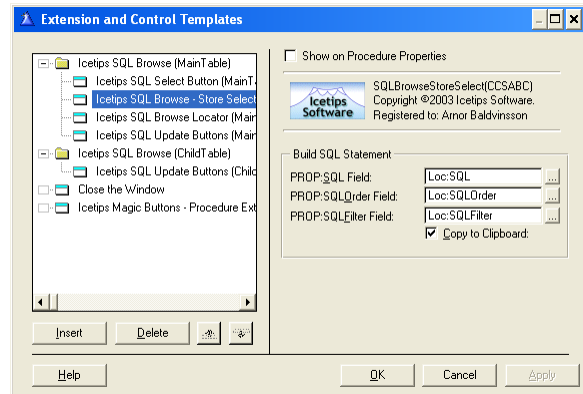
Procedure Extension Templates

Null fields before Adding Record

This template sets all blank or 0 fields in the selected file to NULL before the record is written to the database. This template adds two embed points, Start of Setnull Routine and End of Setnull Routine.

Store Select Statement

This extension template adds a new method to the SQL class, called ReadSQLProperties. It prompts for variables to store the various SQL properties, like PROP:SQL, PROP:SQLOrder, PROP:SQLFilter etc. If the ReadSQLProperties method is called, the variables will contain the information about the SQL statement sent to the driver and optionally copy the SQL string to the clipboard.



WEB Size Extension

No documentation available at this time.

Security Page Setup

No documentation available at this time.

Routine Declaration

No documentation available at this time.

Calculate monthly loan payment

No documentation available at this time.

Restore Child After Cancel

This template was provided as an addition by Horizon Business Concepts, Inc. It is provided as is for those who may need it. It is to be used on forms that have an Icetips SQL browse on a child file. If records have been added to the browse when the form is in insert mode and then the form is cancelled, these records would be orphans. This template takes care of that and deletes the orphan records from the child table. At this time we do not have any further documentation about this extension template.

Code and working examples

In this section of the manual we will provide some code examples from our own applications that might help users to find solutions to common questions. This should not be seen as a complete guide in any way, only a few samples from working applications. The examples are in no special order in this manual. We will keep adding to this and we encourage you to send us code examples, that you have found useful.

Reset browse based on value from a dropdown

In many cases you may need to reset a browse after something has happened, for example if you are selecting a parent value from a dropdown, you need to reset the browse if the value changes. In this example, the dropdown is an ABC dropdown with only a few records in it and the SQL browse shows accounts that belong to the broker from the ABC dropdown. Obviously the variable used in this code must be a hot field in the dropdown. The code is very simple and goes into the Accepted event embed on the dropdown:

```
SQL1.SetFilterScope(HPROP:FilGlobal)
SQL1.SetGlobalFilter('ACC."BrokerID" = ' & Loc:BrokerID)
SQL1.Reset(1)
```

To start the browse using the same criteria, you need this same code in the code section of the TemplateAutoInit method, priority 5001, except you do not call the Reset method. To make this manageable, create a routine, for example:

```
NewBrokerSelected          ROUTINE
  SQL1.SetFilterScope(HPROP:FilGlobal)
  SQL1.SetGlobalFilter('ACC."BrokerID" = ' & Loc:BrokerID)
```

Now you can call this routine from TemplateAutoInit without change, and in the Accepted event on the dropdown you change the code to:

```
Do NewBrokerSelected
SQL1.Reset(1)
```

Child browse on parent form

This is very simple to do and requires no handcode. Right click on the child browse, select Actions and then click on the "Browse Box Behavior" button and go to the "Filter" tab. Click on the "Global Filter" button and in the "Global filter/range limit:" entry field, type in something like this:

```
!'TRN."IOISysID" = ' & IOI:TransactionID
```

Note the starting exclamation mark, which means that the rest of the field is used exactly as it is typed in. In this case the listbox is range limited to the value of IOI:TransactionID, which can have multiple transactions related to it. Translated to Clarion, this filter is the same as TRN:IOISysID = IOI:TransactionID.

Synchronizing multiple child browses

On complex windows, it may be required to synchronize many child browses to one parent browse. This may also need to be conditional for example based on what tabs are active etc. In the following example, a browse has two child browses. We do this on the parent browse, in the FetchQueue method. You will need to add code both into the Data and Code sections. Let's start with the Code section:

```
SynchFilterMulti CString(10000)
```

We could do this dynamically, but in this case we are just happy with 10,000 bytes and it is always going to be more than enough for this particular case.

Now let's see what we have in the Code section and then take a closer look. ?TradeIOISheet is the sheet where the two child browses are on two separate tabs. ?TradeSheet is a sheet with two tabs that determines a global filter for the SQL11 browse. SQL10 is the parent browse (ACC - accounts), SQL11 (TRN - Transactions) and SQL12 (IOI - Indications Of Interests) are the child browses.

```
GET(SELF.Q,xSel)
IF ~ERRORCODE() THEN
  SynchFilterMulti = ''
  Case Choice(?TradeIOISheet)
  Of 1
    Case Choice(?TradesSheet)
    Of 1
      SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:BrokerID') &|
        ' = ' & SQL10:Q.ACC:BrokerID
      SynchFilterMulti = SynchFilterMulti & ' AND '
      SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:AccountID') &|
        ' = ' & SQL10:Q.ACC:AccountID
      SQL11.ForceGlobalFilter(SynchFilterMulti,True)
    Of 2
      SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:BrokerID') &|
        ' = ' & SQL10:Q.ACC:BrokerID
      SynchFilterMulti = SynchFilterMulti & ' AND '
      SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:ContactID') &|
        ' = ' & CON:ContactID
      SQL11.ForceGlobalFilter(SynchFilterMulti,True)
    End
  Of 2
    SynchFilterMulti = ''
    SynchFilterMulti = SynchFilterMulti & SQL12.GetFieldsSQLName('IOI:BrokerID') &|
      ' = ' & SQL10:Q.ACC:BrokerID
    SynchFilterMulti = SynchFilterMulti & ' AND '
    SynchFilterMulti = SynchFilterMulti & SQL12.GetFieldsSQLName('IOI:AccountID') &|
      ' = ' & SQL10:Q.ACC:AccountID
    SQL12.ForceGlobalFilter(SynchFilterMulti,True)
  End
  RETURN True ! This return is VERY important, without Edit/Delete will not work!
ELSE
  SynchFilterMulti = '1 = 2'
  SQL11.ForceGlobalFilter(SynchFilterMulti,True)
  SynchFilterMulti = '1 = 2'
  SQL12.ForceGlobalFilter(SynchFilterMulti,True)
  RETURN True
END
RETURN False
```

The first tab, which contains the TRN browse can be filtered either by the AccountID, or by the ContactID, showing either transactions related to this account, or transactions related to the contact, the owner of the account.

```

Case Choice(?TradesSheet)
Of 1
  SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:BrokerID') &|
    ' = ' & SQL10:Q.ACC:BrokerID
  SynchFilterMulti = SynchFilterMulti & ' AND '
  SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:AccountID') &|
    ' = ' & SQL10:Q.ACC:AccountID
  SQL11.ForceGlobalFilter(SynchFilterMulti,True)
Of 2
  SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:BrokerID') &|
    ' = ' & SQL10:Q.ACC:BrokerID
  SynchFilterMulti = SynchFilterMulti & ' AND '
  SynchFilterMulti = SynchFilterMulti & SQL11.GetFieldsSQLName('TRN:ContactID') &|
    ' = ' & CON:ContactID
  SQL11.ForceGlobalFilter(SynchFilterMulti,True)
End
  
```

We build up the filter by appending the SQL name of the parent field, in the first case, TRN:BrokerID by calling the GetFieldsSQLName method and on the right side of the equation is the value of the queue field from the Accounts browse. These queue fields are always constructed in the same way: The classname for the browse, which you can see if you open the actions for the browse as it is displayed on the main button. The next part is :Q. which is the name of the queue that the templates create - it does not change. After that is the fieldname from the dictionary. So if you need to refer to MYF:SystemID and your SQL class is called SQL7, it would be:

```
SQL7:Q.MYF:SystemID
```

So it is not all that difficult to work with, just learn and remember the basic rules about how it is constructed.

Next step, we add an AND to the SQL statement as we are doing a filter on two fields. Then we build up the second part of the statement in exactly the same way as the first one and when we are done. Now we can call the ForceGlobalFilter and pass it the string with the SQL statement that we have been building up. Note that the second parameter to ForceGlobalFilter is redundant and can be omitted.

Same thing with the second tab, but there we use a field (CON:ContactID) that is not in the queue, but directly in a file that is active at this point.

Notice the bold line with the RETURN True. That RETURN is VERY important because without it the code falls to the bottom and returns False, which means that the method thinks it didn't work and couldn't find the right record in the queue, and will not allow you to update or delete from the browse!

Filter using an Option and Radio buttons

In many cases it is convenient to range limit browses by using options and radio buttons. These are easily selected by the user and convenient to use if the data set is completely fixed, for example to show paid or unpaid transactions etc. In the example below it is actually used to range limit a browse based on parent ID, grand parent ID or great grandparent ID!

The Loc:TRNFilter is a local byte variable, with "This Indication|Account|All Accounts" in the Validity checks - must be in list and "0|1|2" as values.

```
SQL6.SetFilterScope(HPROP:FilGlobal)
Case Loc:TRNFilter
Of 0
    SQL6.SetGlobalFilter('TRN."BrokerID" = ' & IOI:BrokerID &|
                        ' And TRN."IOISysID" = ' & IOI:TransactionID)
Of 1
    SQL6.SetGlobalFilter('TRN."BrokerID" = ' & IOI:BrokerID &|
                        ' And TRN."AccountID" = ' & IOI:AccountID)
Of 2
    SQL6.SetGlobalFilter('TRN."BrokerID" = ' & IOI:BrokerID &|
                        ' And TRN."ContactID" = ' & IOI:ContactID)
End
SQL6.Reset(1)
```

Class reference

The Icetips Cowboy SQL is based on the CCSSql1 class, which is derived from the CCSButtons class. The classes are stored in ccs*. * files which are installed into the LibSrc directory for Clarino 5.5/Clarion 6.0. The following class reference details each property and each method in all the classes in the Icetips Cowboy SQL product. Some of these do not have any explanation and this lists both private and protected properties. We will continue to improve on this class reference in future versions.

Please note that we use similar color coding as the Clarion IDE. Properties that are **protected are red**, **private are maroon** and **virtual methods are green**.

CCSSQL1 Class

Class Properties

ACCESS	&FILEMANAGER
Description:	FileManager instance that is used in the class.
ACTIVEINVISIBLE	BYTE
Description:	Property that determines if the browse is refreshed when it is not visible.
ALERT	LONG(0)
Description:	Browse reorder (header click) hot key.
ALLOWSWAP	BYTE(TRUE), PROTECTED
Description:	Flag to indicate if column swapping is ON or OFF
AQ	&PARAMQ, PROTECTED
Description:	Joined to PQ thru ID, references to parameters for procedure calls. ParamQ is defined as:
	<pre> ParamQ QUEUE, TYPE ID STRING(10) ParamNum BYTE(0) ParamRef ANY END </pre>

ASCENDINGORDERSIGN	STRING(1)
Description:	One character string that contains the character used to indicate ascending order. Default value is plus (+)
BACKENDDATEFORMAT	CSTRING(20)
Description:	Date format required by backend queries.
BOF	BYTE(1), PROTECTED
Description:	True or False for Beginning Of File was reached.
BROWSESTARTED	BYTE, PROTECTED
Description:	Flag True/False that at least 1 call to ApplyOrder() has been made.
BUFFERCOLFILTER	&STRING, PROTECTED
Description:	Column filter buffer. Dynamically created and destroyed.
COLSNAPTHRESHOLD	SHORT(10)
Description:	Affects column resizing when sort orders are changed. Default value is 10.
COLUMNMESSAGE	BYTE(0), PROTECTED
Description:	Indicates if a message should be displayed when the user clicks on the header of a column that can not be sorted. True if enabled, False if disabled
COLUMNORDERTYPE	BYTE(HPROP:COLUMNORDERBY), PROTECTED
Description:	HPROP:ColumnOrderBy or HPROP:NoColumnOrderBy
COUNTRESETSTART	LONG, PRIVATE
Description:	Debug purposes.
DESCENDINGORDERSIGN	STRING(1)
Description:	One character string that contains the character used to indicate descending order. Default value is minus (-)
DOWNFETCHROWS	LONG(5), PROTECTED
Description:	Extra rows fetched when Down Arrow is pressed. Default is 5 rows at a time.

DRAGCOL **LONG(0), PROTECTED**

Description: Column in which EVENT:Drag was started.

EOF **BYTE(0), PROTECTED**

Description: True or False for End Of File reached.

EQ **&EXTSQLQ, PROTECTED**

Description: References to extending SQL statements to be concatenated as indicated by Pos. The ExtSQLQ is declared as:

```

ExtSQLQ      QUEUE, TYPE
Pos          BYTE(0)  !HPROP:BeforeFrom to
              !HPROP:Replace
SQLString    &CSTRING !Reference to string
              !containing extended SQL.
              END
    
```

FETCHRECORDONSELECT **BYTE(FALSE), PROTECTED**

Description: Toggles auto fetch record when selected in browse.

FIELDS **LONG(0), PROTECTED**

Description: Number of records in FQ

FILLFORWARD **BYTE(1), PROTECTED**

Description: Flag to indicate fill direction forward or backward.

FILSCOPE **BYTE(HPROP:FILNONE), PROTECTED**

Description: Filter Scope.

FIRSTSORTCOL **USHORT, PROTECTED**

Description: First sortable column (reading from left to right).

FQ **&COLUMNQ, PROTECTED**

Description: References to file fields and queue fields (queue fields only). The ColumnQ is declared as:

```

ColumnQ     QUEUE, TYPE
Col         USHORT(0)  !Column number being
              ! described
FName       STRING(50) !Unique ID of this column
              ! (Full Field Name)
FFld        ANY        !Reference to file field
QFld        ANY        !Reference to queue field
Color       LONG(0)    !Column color at startup
Width       LONG(0)    !Column width at startup
CaretWidth  LONG(0)    !Column width with carets
    
```

```

NoSort          BYTE(0)      !If true, disallow click-
                ! header resort on this
                ! column (hence, no locate).
Forced          BYTE(0)      !If true, this is a forced
                !field.
FcdFld         ANY          !Reference to browse queue
                ! fill field if forced.
FilsQL         &STRING      !SQL style filter
Order          BYTE(HPROP:Ascending) !Current order of
                ! this column
Format         STRING(200)  !Column format string
AltName        STRING(50)   !Column name when a forced
                ! column (for queue sorting)
Header         STRING(50)   !Column header at startup
CaretHeader    STRING(50)   !Column header with carets
Picture        STRING(50)   !Column picture at startup
QueueFieldNr   Short       !Field number in Queue -
                !used to move columns around
                END

```

GLOBALFILTER	&CSTRING
---------------------	---------------------

Description: Filter applied globally across all columns.

GLOBALFILTERBUFFER	&CSTRING, PRIVATE
---------------------------	------------------------------

Description: Used internally to buffer the global filter to monitor parent browse reset.

GROUPLIST	&CSTRING
------------------	---------------------

Description: String created as needed when select list contains functions.

HEADERSPREPARED	BYTE
------------------------	-------------

Description: Used to determine if the headers have been prepared or not. This is used in the SetHeader method. It is set to False in the Init method and to True in the UpdateHeaders method.

ININAME	STRING(255), PROTECTED
----------------	-------------------------------

Description: Name of application INI file.

INSTANCENAME	STRING(30)
---------------------	-------------------

Description: Name of this instance for INI writes (Uses string version of object name.)

JOINTYPE	BYTE(HPROP:JOINOUTER), PROTECTED
Description:	The type of join, can be either HPROP:JoinInner for inner joins, or HPROP:JoinOuter for outer joins.
LASTCOL	LONG(0), PROTECTED
Description:	Copy of last column clicked for reversion. Default is left hand column, column 1.
LASTSELECT	&CSTRING
Description:	Last select statement.
LASTVSCROLLPOS	LONG(0), PROTECTED
Description:	Most recent position of the vertical scroll thumb.
LIST	LONG(0), PROTECTED
Description:	The browse control (list box control)
LOADANDORDERBYTYPES	BYTE(HPROP:PAGELOAD+HPROP:COLUMNORDERBY), PROTECTED
Description:	Flags how browse is loaded (HPROP:PageLoad or HPROP:FullLoad) plus how browse is ordered (HPROP:ColumnOrderBy or HPROP:NoColumnOrderBy) HPROP:NoColumnOrderBy in this setting causes column headerclicking to be disabled
LOADPENDING	BYTE
Description:	Not used.
LOADTYPE	BYTE(HPROP:PAGELOAD), PROTECTED
Description:	Determines if the browse is page loaded or file loaded. Default is page loaded. Available values are HPROP:PageLoad or HPROP:FullLoad.
LOC	&STRING, PROTECTED
Description:	Locator variable.
LOCCASE	LONG, PROTECTED
Description:	Case of original locator is buffered here and reapplied by SetLocatorPic()
LOCLEARCTL	LONG(0), PROTECTED
Description:	FEQ of the Locator clear button.

LOCCTL	LONG(0), PROTECTED
Description:	FEQ of the Locator control.
LOCFILTER	&CSTRING
Description:	Applied with any other filter on locate.
LOCMGR	&CCSLOCMANAGER
Description:	Class that manages which browse owns the locator. See the CCSLocManager documentation on page 86.
LOCPROMPTCTL	LONG(0), PROTECTED
Description:	FEQ of the Locator prompt control.
LOCSHARED	BYTE(0)
Description:	Flags locator shared, True or False.
LOCTYPE	BYTE(HPROP:LOCSIMPLE), PROTECTED
Description:	Determines the Locator type, simple or progressive. Values can be either HPROP:LocSimple or HPROP:LocProgressive.
MARK	&BYTE
Description:	Reference to queue marker (stored in browse queue).
OBJECTNAME	CSTRING(31)
Description:	Contains the name of the class object. Used in calls to wDebug.
PAGESAHEAD	LONG(0), PROTECTED
Description:	Buffer setting.
PAGESBEHIND	LONG(0), PROTECTED
Description:	Buffer setting.
PAGESIZE	LONG(1), PROTECTED
Description:	Buffer setting.
POSTFIXSIGN	STRING(1)
Description:	One character string that determines what to use as a postfix to the header of the current sort column. Default value is >

PQ **&PROCQ, PROTECTED**

Description: Queue of stored procedures. The ProcQ is declared as:

```

ProcQ      QUEUE, TYPE
ID         STRING(10)
Direction  BYTE(0)      !HPROP:FillForward or
                        !HPROP:FillBackward
FName      STRING(50)   !Used to find FQ record by
                        !FieldName when stored
                        !procedure.
Proc       STRING(100)
                        END
    
```

PREFIXSIGN **STRING(1)**

Description: One character string that determines what to use as a prefix to the header of the current sort column. Default value is <

PRIMARY **&FILE**

Description: Primary view file.

PRIMARYALIAS **CSTRING(20)**

Description: Primary view file alias.

PROCNAME **STRING(30), PROTECTED**

Description: Name of the procedure for INI writes.

Q **&QUEUE**

Description: Browse Queue reference.

QRECS **LONG(0), PROTECTED**

Description: Records in browse Queue from RECORDS(SELF.Q)

RELATIONS **&CSTRING**

Description: String contains AddRelation() created relations.

RESET **BYTE(0)**

Description: Flag to indicate if the browse should be reset.

RESETS **&FIELDPAIRSCLASS**

Description: Instance of the FieldPairsClass that is used to determine if a value of a field that has been registered has changed.

RESETWHENVISIBLE	BYTE
Description:	Determines if the browse needs to be reset when it becomes visible. Internal use only.
ROWS	LONG(0), PROTECTED
Description:	Number of rows to load into listbox.
SAVEFORMAT	BYTE(TRUE), PROTECTED
Description:	Flag to indicate if the browse format should be saved. Set to True or False to turn it ON or OFF.
SEL	LONG(0), PROTECTED
Description:	Currently selected queue record.
SESSIONTIME	LONG, PRIVATE
Description:	For debugging.
SORTCOL	LONG(1), PROTECTED
Description:	Currently active sort order column default is left hand column (1)
SORTCOLOR	LONG(0), PROTECTED
Description:	Color of sorted column when active.
SQ	&SELECTQ, PROTECTED
Description:	References to select statement fields (all fields in select statement)
SYNCHCHILDID	STRING(30), PROTECTED
Description:	Name of child field in parent/child join relationship.
SYNCPARENTID	ANY, PROTECTED
Description:	Reference to parent field in parent/child join relationship.
TABLENAMEQ	&TABLENAMEQUEUE
Description:	A reference to the TableNameQueue. This queue is not used in the classes.
TIMEOUT	LONG(0), PROTECTED
Description:	Buffer setting.

TOOLBAR	&TOOLBARCLASS
Description:	A reference to the ToolbarClass ABC class.
TOOLBARITEM	&CCSTOOLBARLISTBOXCLASS
Description:	A reference to the CCSToolBarListBoxClass class. Please refer to page 88 for more information about this class.
TOOLCONTROL	SIGNED
Description:	Not used by the class.
USEFILEPREFIX	BYTE(0)
Description:	Not used by the class.
VIEWOPEN	BYTE(0), PROTECTED
Description:	Indicates if the View is Open or not.
VIEWPARAM	BYTE(0), PROTECTED
Description:	Parameter count for this procedure.
VIEWPOS	&STRING
Description:	Reference to file view position (stored in browse queue)
VIEWPROC	BYTE(0), PROTECTED
Description:	Flag indicates use current view procedure.
VPos	LONG(0), PROTECTED
Description:	Vertical position offset.
Vw	&VIEW
Description:	Browse view reference.
WHERECLAUSE	&CSTRING
Description:	String created as needed or referencing user-owned where expression.
WINDOW	&WINDOWMANAGER
Description:	Reference to the Window Manager ABC object.

Class Methods

ADDFIELDFUNCTION	LONG, PROC
-------------------------	-------------------

Parameters: String xFldName
 String xFunc

Return type: Long

Description: This method adds a call to an SQL function as specified in xFunc to fill the Field specified in xFldName.

ADDQFROMBUFFER	VIRTUAL
-----------------------	----------------

Parameters: No parameters

Return type: Does not return a value

Description: Add a new browse Q record from the record buffer and store it's file position marker.

ADDQUEUEFIELD

Parameters: Long xCol
 String xFName
 *? XFld
 *? XQFld
 Byte xNoSort
 <*? xFcdFld>
 <String xAltName>

Return type: Does not return a value

Description: Establish information about the queue fields that will be loaded from the file. Properties such as header,width, format which are likely to change are buffered in the queue. The field's column position in the browse is the queue key.

ADDERELATION

Parameters: String xPrefix
 *KEY xKey
 *Group xRec
 String xFldList

Return type: Does not return a value

Description: In the event the programmer provides no WHERE clause in the template, the template calls here. This code figures out the relationships between the tables, given the keys and a list of fields supplied by the template (xFldList).

ADDRESETFIELD

Parameters: *? Field

Return type: Does not return a value

Description: Adds the passed field/variable to the Resets property which is an instance of the ABC FieldPairsClass.

ADDSELECTFIELD

Parameters: Long xSelPos
String xFName
* ?Fld
File xFile
*Group xRec
FileManager xFM
<*Key xSelKey>
<String xKeyName>
String xFilePrefix

Return type: Does not return a value

Description: Establish information about the SELECT fields comprising the browse. Some of this could be gotten elsewhere with property inspection, but this brings everything to one place and makes the class less sensitive to driver differences. SQ stands for SelectQ. Inner-style SELECT statements are built with the field information stored here. The field's Clarion name (including prefix) is the queue key. The driver's file alias property is set here, based on the Clarion prefix.

Date and time fields have to be handled differently when they're in a group as this indicates they're being used to define an SQL timestamp. This is a typical structure:

```
pubdate          STRING(8),NAME('pubdate')
pubdate_GROUP   GROUP,OVER(pubdate),NAME('pubdate_GROUP')
pubdate_DATE    DATE
pubdate_TIME    TIME
                END
```

In this situation you can't use the DATE or TIME field in the select statement because those fields only exist in the Clarion record structure. Instead you locate the name of the field OVER which the group has been declared.

ADDTOOLBARTARGET

Parameters: ToolbarClassT

Return type: Does not return a value

Description: Clone of the ABC version of the same name. Used to hook the toolbar into the browse and establish an instance of CCSToolBarListBoxClass. CCSToolBarListBoxClass only differs from the ABC one in that it does not use a reference to the browse, only the list box FEQ to determine whether the browse is visible.

ADDVIEWPROCEDURE **LONG, PROC**

Parameters: String xProcID
 String xProcCall
 Byte xRslt
 <String xColVar>
 <Byte xDir>

Return type: Long, Proc

Description: Add a stored procedure into a bank of callable stored procedures. ! PQ stands for ProcedureQ. Stored procedures are stored by a programmer-determined ID (xProcId). This can be any unique string by which the procedure can be identified when needed.

ADDVIEWPARAMETER **BYTE, PROC**

Parameters: String xProcID
 Byte xParamNum
 *? xParamRef

Return type: Byte, Proc

Description: Adds an new parameter to an installed, stored procedure identified by xProcId, and xParamNum. If a parameter of number xParamNum already exists, this replaces it.

ALLOWSWAPPING

Parameters: Byte xYesNo

Return type: Does not return a value

Description: Determines if the browse allows columns to be swapped. XYesNo parameter can be True or False.

APPLYORDER **VIRTUAL**

Parameters: Byte xForce

Return type: Does not return a value

Description: Virtual method. Master method called by the template to start a browse off. If xForce is stipulated the browse is immediately filled with data from the file. If xForce is omitted the browse has all the necessary properties set - width, color, header of the sort column. The first time called at browse startup the browse format strings are read from the INI. and the SELF.BrowseStarted flag is set to true.

This method can also be called any time with xForce to refresh the browse. Call with xForce set to False if you want to initialize a child browse which should not be immediately filled till the parent browse forces a global filter on it.

BUFFER **VIRTUAL**

Parameters: <Long xPg>
 <Long xBehind>
 <Long xAhead>
 <Long xTime>

Return type: Does not return a value

Description: Virtual method. This method is called to initialize the Driver buffer settings. See Clarion documentation about the Buffer statement.

BUFFERGLOBALFILTER

Parameters: No parameters

Return type: Does not return a value

Description: Stores a copy of the current global filter into the SELF.GlobalFilterBuffer property.

BUILDEXTENDEDSQL **STRING, VIRTUAL**

Parameters: Byte xPos

Return type: String

Description: Virtual method. Returns the Extended SQL addins determined by the programmer in the template. xPos can be any one of the flags defined in CCSSQL1.inc for BuildExtendedSQL:

```

HPROP:BeforeFrom      EQUATE (1)
HPROP:BeforeWhere     EQUATE (2)
HPROP:BeforeOrderBy   EQUATE (3)
HPROP:AfterOrderBy    EQUATE (4)
HPROP:Replace          EQUATE (5)
HPROP:ReplaceOrderBy  EQUATE (6)
  
```

BUILDFIELDLIST **STRING, VIRTUAL**

Parameters: No parameters

Return type: String

Description: Virtual method. Builds a list of fields determining the field SELECT list into an SQL statement. If there's a field function in the select statement this function creates the GROUP BY component required. Any extended SQL of type HPROP:BeforeFrom, is installed here. The programmer is responsible for making sure his SQL component is correctly formatted given its placement.

BUILDFILELIST **STRING, VIRTUAL**

Parameters: No parameters

Return type: String

Description: Virtual method. Builds the view's file list into SQL including the file aliases determined earlier from Clarion prefixes. If there's any extended SQL (of type HPROP:BeforeWhere) established by the programmer in the template, this is appended to the file list. The programmer is responsible for ensuring that the correct conjunction is present.

BUILDORDERSTATEMENT STRING, VIRTUAL

Parameters: String xFName
Return type: String
Description: Virtual method. Master call for building the order statement. A good place to embed code if you want to build the Order clause.

BUILDSELECTSTATEMENT STRING, VIRTUAL

Parameters: No parameters
Return type: String
Description: Virtual method. Builds the final-final select statement, taking into account all of the probable settings. Individual sections are commented below.

BUILDVIEWPROCEDURE STRING, VIRTUAL

Parameters: No parameters
Return type: String
Description: Virtual method. Builds the view procedure components into SQL, ready to be sent to the data base.

BUILDWHERESTATEMENT STRING, VIRTUAL

Parameters: No parameters
Return type: String
Description: Virtual method. Builds the final-final WHERE statement.

CALLUPDATE LONG, PROC, VIRTUAL

Parameters: Long xRequest
Return type: Long, Proc
Description: Virtual method. Placeholder method into which the template can write embeds with the relevant update form.

CHANGE BYTE, PROC, VIRTUAL

Parameters: Long xSel
Return type: Byte, Proc

Description: Virtual method. This method is called by the change button accepted event.

CHECKLOCATORFILTER **STRING**

Parameters: String pFilter

Return type: String

Description: This function doubles up single quotes in the locator text. This prevents problems with the backends when locating for example on the word **Sam's** and it is changed to **Sam''s**

CHECKRESETFIELDS

Parameters: No parameters

Return type: Does not return a value

Description: This method uses the EqualBuffer method of the FieldPairsClass to detect changes in any registered reset fields. If the fields have changed. The new value is assigned to the FieldPairsClass to prevent multiple resets and the browse is forced to reset if needed.

CLEARCURRENTFILTER

Parameters: No parameters

Return type: Does not return a value

Description: This method clears the Global or Column filter based on what filter scope is active. It does not reset the browse or change the scope.

CLEARGLOBALFILTER

Parameters: <Byte xForce>

Return type: Does not return a value

Description: Clears the global filter. If xForce is true, it forces a browse refresh. Otherwise there is no effect until some other cause, such as a header click forces a browse refresh.

CLEARGLOBALFILTERBUFFER **VIRTUAL**

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Clears the buffered version of the global filter.

CLEARLOCATOR **LONG, PROC, VIRTUAL**

Parameters: No parameters

Return type: Long, Proc

Description: Virtual method. Clears the locator and resets the browse but only if this browse owns the locator (if it's shared). Non-shared locators are obviously always owned by the browse to which they are initialized.

CLEARLOCATORNOFORCE VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Clears the locator but does not immediately refresh the browse.

CLEARVIEWPROCEDURE VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Clears the ViewProc Flag (True/False) which, if true, causes SELF.ResetQueue() and SELF.Reset(True) to refresh from the currently selected stored procedure.

CLEARWHERECLAUSE VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Clears the current value in SELF.WhereClause

CLOSEVIEW VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Closes the view (if it is open) and clears any view properties.

CONSTRUCT PROTECTED

Parameters: No parameters
Return type: Does not return a value
Description: Protected method. This is the initial constructor and it creates new instances of some property queues to make sure they are ready when the class is initialized and starts.

COUNTVIEWPARAMETERS BYTE, PROC, VIRTUAL

Parameters: String xProclD
Return type: Byte, Proc

Description: Virtual method. This counts the number of parameters that were described as belonging to the installed, stored procedure identified by xProclD.

DELETE **BYTE, PROC, VIRTUAL**

Parameters: Long xSel
Return type: Byte, Proc
Description: Virtual method. Called by the delete button accepted event.

DELETEVIEWPROCEDURE **BYTE, PROC, VIRTUAL**

Parameters: String xProclD
Return type: Byte, Proc
Description: Virtual method. Deletes an installed view procedure.

DELETEVIEWPARAMETER **BYTE, PROC, VIRTUAL**

Parameters: String xProclD
Return type: Byte, Proc
Description: Virtual method. Deletes a parameter from the installed, stored procedure identified by xProclD.

DESTRUCT **PROTECTED**

Parameters: No parameters
Return type: Does not return a value
Description: Protected method. Final destruction of all data created (NEWed) anywhere by this class.

DRAG **PRIVATE**

Parameters: Long xSourceCol
 Long xDestCol
Return type: Does not return a value
Description: Private method. Called by SELF.DropColumn() the determine positions of source and target columns.

DROPCOLUMN **VIRTUAL**

Parameters: No parameters , Virtual
Return type: Does not return a value
Description: Entry point for chain of calls required to drop a column that is being "dragged".

EMBEDSYNCHFILTER
VIRTUAL

Parameters: *CString xFilter

Return type: Does not return a value

Description: Virtual method. Placeholder method into which the template can write embeds.

FETCHFORMAT
LONG, PROC, VIRTUAL

Parameters: No parameters

Return type: Long, Proc

Description: Virtual method. Fetches the current browse formats strings saved in the INI file. The SELF.SaveFormat property must be set to True (Default).

FETCHQUEUE
LONG, PROC, VIRTUAL

Parameters: Long xSel

Return type: Long, Proc

Description: Virtual method. A placeholder method that fetches the queue record specified in xSel. The template writes code in here to send refresh commands to synchronized child browses, if any.

FETCHRECORD
BYTE, PROC, VIRTUAL

Parameters: Long xSel

Return type: Byte, Proc

Description: Virtual method. Fetches the selected record depending on the setting of SELF.FetchRecordOnSelect. If True, regets the record from disk and refreshes this to the Q (and obviously the BUFFER). If False, refreshes the record from Q to BUFFER.

FILLQUEUEFIELD
BYTE, PROC, VIRTUAL, PRIVATE

Parameters: No parameters

Return type: Byte, Proc

Description: Virtual and Private method. A method used internally to pass a data field to a queue field. In the case of a forced sort, where the programmer has control of what goes into the queue field, this method calls into another method of the same name (but with parameters). The programmer can embed into that method. (See below).

FILLQUEUEFIELD
BYTE, PROC, VIRTUAL

Parameters: String xFName
ANY xValue

Return type: Byte, Proc

Description: Virtual method. A placeholder method into which the programmer (with help from the template) can control data going into a queue field. This happens in the case of a "Forced Sort" where the queue displays a local variable filled with data from the file.

FORCECOLUMNFILTER LONG, PROC, VIRTUAL

Parameters: String xFilter

Return type: Long, Proc

Description: Virtual method. Same as the SetColumnFilter method except it forces a browse refresh and sets the FilterScope property automatically.

FORCEGLOBALFILTER

Parameters: String xFilter
<Byte xSel>

Return type: Does not return a value

Description: Same as SetGlobalFilter except that this one causes the browse to instantly refresh. This is used to synchronize child browses. In that case the template calls here from an embed in the FetchQueue method. Please note that the omittable xSel parameter is never used in this method.

FORCEHEADERCLICK LONG, PROC, VIRTUAL

Parameters: String xFname

Return type: Long, Proc

Description: Virtual method. This handles the details when the user clicks on a header column.

FORCEVIEWPROCEDURE

Parameters: String xProcID

Return type: Does not return a value

Description: Forces the browse to refresh using the view procedure identified by xProcID.

GETACTIVEINVISIBLE BYTE

Parameters: No parameters

Return type: Byte

Description: Returns the value of Self.ActiveInvisible

GETCOLUMNFIELD	STRING, VIRTUAL
-----------------------	------------------------

Parameters: No parameters

Return type: String

Description: Virtual method. Probably redundant but still used by the Locate function to determine the SQL field name in building a locator filter.

GETCOLUMNFILTER	STRING, VIRTUAL
------------------------	------------------------

Parameters: No parameters

Return type: String

Description: Virtual method. Returns the column filter string for the current column.

GETCOLUMNFILTER	STRING, VIRTUAL
------------------------	------------------------

Parameters: Long xCol

Return type: String

Description: Virtual method. Returns the column filter string for a specific column (xCol).

GETCOLUMNFROMNAME	LONG, PROC, VIRTUAL
--------------------------	----------------------------

Parameters: String xColVar

Return type: Long, Proc

Description: Virtual method. Returns the column number given the Clarion field name (expressed as a string).

GETCOLUMNHEADER	STRING, VIRTUAL
------------------------	------------------------

Parameters: No parameters

Return type: String

Description: Virtual method. Return the current column header name.

GETCURRENTFILTER	STRING, VIRTUAL
-------------------------	------------------------

Parameters: No parameters

Return type: String

Description: Virtual method. Returns the entire filter, based on how SELF.FilterScope is set.

GETCURRENTORDERSUFFIX	STRING, VIRTUAL
------------------------------	------------------------

Parameters: No parameters

Return type: String

Description: Virtual method. Determines the Order suffix based on fill direction and order properties.

GETDATETIMEFIELDNAME **STRING, VIRTUAL**

Parameters: *FILE xFile
*GROUP xRec
String xFName

Return type: String

Description: Virtual method. Gets the name of the field that is overed by a group containing the field passed as xFName.

GETDEFAULTORDER **LONG, VIRTUAL**

Parameters: No parameters

Return type: Long

Description: Virtual method. Figures out what the default order direction is when a column has a key.

GETFIELDSQLNAME **STRING, VIRTUAL**

Parameters: String xFldName
<String Origin>

Return type: String

Description: Virtual method. Creates a field's SQL name from it's Clarion name, including double quotes.

GETFILTERSCOPE **BYTE, VIRTUAL**

Parameters: No parameters

Return type: Byte

Description: Virtual method. Returns the value of SELF.FilScope.

GETFIRSTSORTCOLUMN **USHORT, PRIVATE**

Parameters: No parameters

Return type: Ushort

Description: Private method. Figures out at browse startup which is the first column that allows column sorting and returns that value to the caller.

GETGLOBALFILTER **STRING, VIRTUAL**

Parameters: No parameters

Return type: String

Description: Virtual method. Returns the value of SELF.GlobalFilter or empty string if there is no GlobalFilter.

GETHEADER		STRING
Parameters:	No parameters	
Return type:	String	
Description:	Returns the header of the current sort column.	
GETKEYCOMPONENTSUFFIX		STRING, VIRTUAL
Parameters:	Byte xElement	
Return type:	String	
Description:	Virtual method. Key referred to is always SELF.SQ.Key. Any non-leftmost key component must take on the opposite of it's normal order if the leftmost key component's order is reversed from the key's order.	
GETLOCFILTER		STRING, VIRTUAL
Parameters:	No parameters	
Return type:	String	
Description:	Virtual method. Returns the current locator filter (if any) - i.e. the value of SELF.LocFilter.	
GETNEXTFIELD		STRING, PRIVATE
Parameters:	String xFldList USHORT xNdx	
Return type:	String	
Description:	Private method. Used by AddRelation method to parse the field list supplied to it into individual fields.	
GETORDER		LONG, PROC, VIRTUAL
Parameters:	No parameters	
Return type:	Long, Proc	
Description:	Virtual method. Inspect the order setting on the current column. If no order setting, determine if there is any "natural" order as mandated by a key.	
GETORDERSIGN		STRING, PRIVATE
Parameters:	No parameters	
Return type:	String	
Description:	Private method. Returns the directional sign + or - that should be applied to a browse header given the current sort order assigned. This is determined by the Self.AscendingOrderSign and Self.DescendingOrderSign properties.	

GETORDERSTATEMENT
STRING, VIRTUAL

Parameters: String xFName

Return type: String

Description: Virtual method. Builds the order clause when the browse is page (batch) loaded and browse order must be determined at file level, with an order clause, based on the currently highlighted column and the current browse order setting (HPROP:Ascending or HPROP:Descending).

GETROWS
LONG

Parameters: No parameters

Return type: Long

Description: Inspect the property that determines the number of rows that are added to a browse each time the ResetQueue method is called.

GETSORTCOLOR
LONG

Parameters: No parameters

Return type: Long

Description: Returns the SortColor property. (The color designated to be applied to the text in the current sort column).

GETSORTCOLUMN
LONG, VIRTUAL

Parameters: No parameters

Return type: Long

Description: Virtual method. Returns the current sort column, i.e. the SortCol property.

GETSORTWIDTH
LONG

Parameters: No parameters

Return type: Long

Description: Returns the width of the current sort column.

GETVIEWFILTER
STRING, PRIVATE

Parameters: No parameters

Return type: String

Description: Private method. Depending on a number of things such as the state of the locator, programmer WHERE clause, column and global filters this determines the pre-final-final WHERE clause.

GETVISIBLE
BYTE

Parameters: No parameters

Return type: Byte

Description: Returns True or False depending if the Prop:Visible is True or False for the listbox control. This is used in implementing the ActiveInvisible property.

HEADERCLICK
LONG, PROC, VIRTUAL

Parameters: Long xKeyCode

Return type: Long, Proc

Description: Virtual method. This handles the details when the user clicks on a header column. Specifics are explained in each of the sections below.

INIT

Parameters: String pObjName

Return type: Does not return a value

Description: This method is responsible for setting up the WindowComponent Interface, set the order signs and sort signs, instantiate the FieldPairsClass and some other initial housekeeping. The name of the class object is passed to it, i.e. 'SQL1' or 'SQL5' for example. This is useful for debugging purposes.

INITLOCATOR

Parameters: *String xLocator
 Long xLocCtl
 Long xLocClearCtl
 Long xLocPrompt
 Byte xLocType
 String xDateForm
 Byte xSharedTF
 CCSLocManager xLocMgr)

Return type: Does not return a value

Description: Initializes a locator for the browse. Includes a reference to the locator variable which is owned by the browse procedure and is determinable by the programmer in the template. The original case (UPPER/LOWER) of the control is trapped as this value seems to be lost when the text property (picture) is changed as different columns are selected to order the browse. If there's any special back-end date format it is provided here and used when the locator filter (WHERE) is created. If the locator is shared, then this browse is added to the LocatorManager object's Q of browses sharing this locator. All locate calls ultimately are directed from the LocatorManager based on the last locator-sharing browse to be selected, re-ordered or CtrlMouseLeft-ed.

INSERT	BYTE, PROC, VIRTUAL
---------------	----------------------------

Parameters: Long xSel
Return type: Byte, Proc
Description: Virtual method. Called by the Insert button accepted event.

INSERTSQL	
------------------	--

Parameters: Byte xPos
String xSQL
Return type: Does not return a value
Description: Inserts user SQL statements into the EQ. (ExtendedSQL Q). All non-legal positional values are ignored.

ISORTCOLUMN	BYTE, PRIVATE
--------------------	----------------------

Parameters: USHORT xCol
Return type: Byte
Description: Private method. Returns true or false to indicate that the column specified in xCol is a sortable column or not.

KILL	
-------------	--

Parameters: No parameters
Return type: Does not return a value
Description: Cleans up hotkeys and popup menu. Also calls the Kill method of the FieldPairsClass and disposes of the FPC instance.

LOCATE	LONG, PROC, VIRTUAL
---------------	----------------------------

Parameters: No parameters
Return type: Long, Proc
Description: Virtual method. Build a locator based on current order column, field picture and whether HPROP:LocSimple or HPROP:LocProgressive. A progressive locator isn't cleared between searches until the user presses the CLR button.

OPENVIEW	VIRTUAL
-----------------	----------------

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Open the view, after first ensuring that is is closed.

POSITIONFQ	LONG, PROC, PRIVATE
-------------------	----------------------------

Parameters: Long xColumn

Return type: Long, Proc
Description: Private method. Positions the field Q to the column supplied in xCol. This exposes all the properties describing the field (See Q definition in the INC file).

POSITIONSQFIELD **LONG, PROC, PRIVATE**

Parameters: String xFldName
Return type: Long, Proc
Description: Private method. Positions the SELECT queue to a specific select field's properties using field name as the retrieval indicator.

POSITIONSQFILE **LONG, PROC, PRIVATE**

Parameters: String xFileName
Return type: Long, Proc
Description: Private method. Positions the SELECT queue to a specific select field's properties using file name as the retrieval indicator.

POSITIONSQPos **LONG, PROC, PRIVATE**

Parameters: USHORT xPos
Return type: Long, Proc
Description: Private method. Positions the SELECT queue to a specific select field's properties using field position as the retrieval indicator.

PRIMERECORD **LONG, PROC, VIRTUAL**

Parameters: Byte SuppressClear=0
Return type: Long, Proc
Description: Virtual method. Calls the PrimeRecord ABC method to prime a record, including auto-inc if indicated in the dictionary. Version 6.000 supports SuppressClear parameter, which is False by default.

READSQLPROPERTIES **VIRTUAL**

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. The ReadSQLProperties virtual is a placeholder for template code that lets the user specify which fields to store the SQL properties in. It must be called any time PROP:SQL, PROP:SQLOrder, or PROP:SQLFilter is set. The derived virtual, if any, can then take action based on the values of these properties. Currently there is an extension to the browse which lets the user store these properties in a variable and optionally copy them to the clipboard.

REFRESHBUFFERFROMQ
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Reads the Browse Q fields into the record BUFFER for the currently selected record.

REFRESHQFROMBUFFER
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Reads the record BUFFER fields into the browse Q.

REGETRECORD
LONG, PROC, VIRTUAL

Parameters: Long xSelected

Return type: Long, Proc

Description: Virtual method. Reget the queue record in xSelected from the file.

REPOSITIONFQ
LONG, PROC, PRIVATE

Parameters: No parameters

Return type: Long, Proc

Description: Private method. Repositions the field Q to the current sort column settings. As field properties are inspected for one reason or another, the field Q position is moved from the current sort column setting. A call here will bring the current SELF.SortCol field properties back into scope.

RESET
VIRTUAL

Parameters: Byte xForce=0

Return type: Does not return a value

Description: Virtual method. Resets the record when xForce is false. (See the FetchRecord method) Resets the browse when xForce is true. (See the ResetQueue method when Reset = True) This method is called by the WindowComponent method calls to reset the browse.

RESETONWINDOWUPDATE

Parameters: No parameters

Return type: Does not return a value

Description: Not implemented in 6.000

RESETQUEUE
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Master method that calls all the relevant methods to reset (Refresh) the browse from the current position, or from the top or bottom, depending on the state of the Reset property (True or False), and the FillForward property (True or False). It also sets various flags to indicate it has reached the top of the data set (SELF.Bof) or the bottom (SELF.Eof).

RESETROWS
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Reset the number of rows added to the browse queue each time the ResetQueue method is called or the Reset(True) method is called back to the number of visible items in the list box. Next browse refresh will be a complete reset.

RESETSORTCOLUMN

Parameters: Byte pColumnNumber

Return type: Does not return a value

Description: Sets the pColumnNumber column the active sort column.

RESETSTART
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. This method is called by SELF.ResetQueue() when the browse Q is empty or when SELF.Reset is set to True. It closes the view and reopens it with whatever new affecting properties are in place. ~~For debugging, this method will write inner selects to a file less cluttered than the standard ODBC trace log. With multiple browses on a window it will also indicate which browse selected what and how many times browse reset was fired on each browse. To turn this on, add the word "DebugOn" into the Project/Properties/Defines area and recompile. (Don't forget to turn it off before you deliver or you'll create a monster.)~~

RESTORECOLUMN
LONG, PROC, PRIVATE

Parameters: No parameters

Return type: Long, Proc

Description: Private method. Restores the list box properties of the column that has lstopped being the sort column. Header changed back, width set back (unless it has been significantly resized by the user), text color

set back to normal. Field Q positioned back to the current sort column.

RESTOREGLOBALFILTER

Parameters: <Byte xForce>
Return type: Does not return a value
Description: Restores the global filter from the temporarily buffered version.

SAVEBROWSEFORMAT

Parameters: Byte xYesNo
Return type: Does not return a value
Description: Sets the SaveFormat property. All out of range values are ignored.

SCROLLDOWN

Parameters: <Long xRows>
Return type: Does not return a value
Description: Scroll down one record. The omissible parameter isn't used here but could be put into use by some embed code in an overriding procedure.

SCROLLUP

Parameters: <Long xRows>
Return type: Does not return a value
Description: Scroll up one record. The omissible parameter isn't used here but could be put into use by some embed code in an overriding procedure.

SELECTBROWSE

LONG, PROC, VIRTUAL

Parameters: <Long xSel>
Return type: Long, Proc
Description: Virtual method. Selects the browse record in the xSel parameter (if any). Also does a fetch record which refreshes the file buffer from the Queue or from the data base depending on the value in FetchOnSelect property, True or False.

SETACTIVEINVISIBLE

Parameters: Byte pActiveInvisible
Return type: Does not return a value
Description: Sets the ActiveInvisible property to True or False.

SETALERT
VIRTUAL

Parameters: Long xAlert

Return type: Does not return a value

Description: Virtual method. Alert the standard hot keys for the browse as well as the configurable hot key SELF.Alert that reorders the browse.

SETCOLUMNFILTER
LONG, PROC, VIRTUAL

Parameters: String xFilter

Return type: Long, Proc

Description: Virtual method. Sets a filter on the current column. Each column in a browse can, in fact have a different filter applied. As the operator changes sort columns with a header click, the data view can change. This setting has no effect unless filter scope is first set with SELF.SetFilterScope(HPROP:FilColumn) and the browse is refreshed. It calls it's namesake stipulating the current sort column.

SETCOLUMNFILTER
LONG, PROC, VIRTUAL

Parameters: String xFilter
Long xCol

Return type: Long, Proc

Description: Virtual method. Sets a filter on a specific column (xCol). Each column in a browse can, in fact have a different filter applied. As the operator changes sort columns with a header click the, data view can change. This setting has no effect unless filter scope is first set with SELF.SetFilterScope(HPROP:FilColumn) and the browse is refreshed.

SETCOLUMNHEADER
LONG, PROC, VIRTUAL

Parameters: String xHdr

Return type: Long, Proc

Description: Virtual method. Change the current column header name.

SETCOLUMNMESSAGE
VIRTUAL

Parameters: Byte xTorF

Return type: Does not return a value

Description: Virtual method. Sets the SELF.ColumnMessage property. All out of range values are ignored.

SETCOLUMNORDERTYPE

Parameters: Byte xType=HPROP:ColumnOrderBy

Return type: Does not return a value

Description: Sets the value of the ColumnOrderType property. If the value is HPROP:NoColumnOrderBy, then all fields in the field queue are marked as "NoSort" and the browse order can not be changed with a click on the browse header. This is irreversible during the life of a browse. The browse would have to be cancelled and restarted to undo HPROP:NoColumnOrderBy. Normally this method is called by the template when the programmer changes certain parameters on the Data Access button in the template.

SETDRAGCOLUMN VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Sets the column number of the column being "dragged" and turns on Clarion Drag and Drop. Since drag and drop is normally hotwired to the mouseleft key, this allows any other (and less commonly used) alert key to begin a drag and drop sequence.

SETFETCHRECORD VIRTUAL

Parameters: Byte xTorF

Return type: Does not return a value

Description: Virtual method. Sets the SELF.FetchRecordOnSelect flag to True or False ignoring all other values.

SETFILLFORWARD VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Sets the FillForward property to true.

SETFILTERSCOPE VIRTUAL

Parameters: Byte xScope
<Byte xForce>

Return type: Does not return a value

Description: Virtual method. Set the filter scope property, ignoring any out-of-bounds values. If the second parameter is nonzero the browse will be immediately refreshed. Otherwise, no refresh happens till some other activity such as a click on the browse header refreshes the browse.

SETGLOBALFILTER VIRTUAL

Parameters: String xFilter

Return type: Does not return a value

Description: Virtual method. Sets the global filter but will have no effect until filter scope has been set to global with

SELF.SetFilterScope(HPROP:FilGlobal) and some other activity such as a header click causes a browse refresh.

SETHHEADER
VIRTUAL

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Sets the header of the current sort column to the header with carets and the appropriate signs to indicate it has become the sort column, by default + or -

SETJOINTYPE

Parameters: Byte xJoinType=HPROP:JoinOuter

Return type: Does not return a value

Description: Sets the SELF.JoinType Flag to inner or outer, ignoring all others.

SETLIST
VIRTUAL

Parameters: Long xList

Return type: Does not return a value

Description: Virtual method. Sets SELF.List property to the list box FEQ in xList.

SETLOADTYPE

Parameters: Byte xType=HPROP:PageLoad

Return type: Does not return a value

Description: Sets the browse load type to HPROP:PageLoad (read: size-defineable batch load) or HPROP:FullLoad (which loads the entire data set to the queue). Load type can be changed at run time. Should be followed by a call to SELF.Reset(True).

SETLOCATOR

Parameters: Long xLocCtl

Return type: Does not return a value

Description: Sets the SELF.LocCtl property to the locator FEQ given in xLocCtl.

SETLOCATORPIC
VIRTUAL

Parameters: <String xPic>

Return type: Does not return a value

Description: Virtual method. Sets the locator picture to match the current sort (search) column.

SETLOCATORTYPE
VIRTUAL

Parameters: Byte xLocType

Return type: Does not return a value

Description: Virtual method. Set the locator type to HPROP:LocSimple or HPROP:LocProgressive. All other values are ignored and default to HPROP:LocSimple

SETORDER
VIRTUAL

Parameters: Long xOrder

Return type: Does not return a value

Description: Virtual method. Changes the order setting on the current column. This method does not force a browse refresh.

SETORDERSIGN

Parameters: String pAsc
String pDesc

Return type: Does not return a value

Description: Sets the Ascending and Descending order signs or characters to use. By default they are set to + and -

SETQUEUE
VIRTUAL

Parameters: *QUEUE xQ
*Byte xMark
*String xPos

Return type: Does not return a value

Description: Virtual method. References the SELF.Q property to the browse Queue specified in xQ. Also references the Queue marker field and the Queue position field.

SETREVERSE

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Toggles the SELF.Order property between HPROP:Ascending and HPROP:Descending. The lagging value, SELF.LastCol ensures that a reversal only happens if you request it of the current sort column. I.E. a column doesn't reverse the first time you click on it, only if you click on it a second time. The first click displays it in its current order state.

SETROWS
VIRTUAL

Parameters: Long xRows

Return type: Does not return a value
Description: Virtual method. Set the property that determines the number of rows that are added to a browse each time the ResetQueue method is called.

SETSCROLLTHUMB VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Positions the vertical scroll bar thumb at top, bottom or in the middle depending on the position in the queue. If at the top, it's at top, if at the end, it's at the bottom, anything else results in the thumb being in the middle.

SETSORTCOLOR VIRTUAL

Parameters: <Long xColor>
Return type: Does not return a value
Description: Virtual method. Sets the color of the text in the current sort column to the color specified in xColor or in the SELF.SortColor property in the event that xColor is omitted.

SETSORTCOLUMN LONG, PROC

Parameters: Long xCol
Return type: Long, Proc
Description: Allows setting of the current sort column property without exposing the property to illegal values. Non-sorting columns are ignored.

SETSORTSIGN

Parameters: String pPrefix
String pPostFix
Return type: Does not return a value
Description: Sets the characters used to indicate that the column is the active sort column. By default < and > are used.

SETSORTWIDTH VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Sets the width of the current sort column to accomodate the carets and the sort order signs (+,-) when a column becomes the sort column.

SETVIEW

Parameters: *View xView
 FILE xPrimary
 String xAlias

Return type: Does not return a value

Description: Virtual method. References the SELF.Vw property to the browse VIEW specified in xView. Also stores the primary file name and it's alias.

SETVIEWPROCEDURE
VIRTUAL

Parameters: String xProcID

Return type: Byte, Proc

Description: Virtual method. Sets the view procedure which will be in effect the next time the browse is refreshed by a call to the ResetQueue method.

SETVIEWPARAMETER
BYTE, PROC, VIRTUAL

Parameters: String xProcID
 Byte xParamNum

Return type: Byte, Proc

Description: Virtual method. Establishes a change to a parameter of an installed, stored procedure identified by xProcID and xParamNum

SETWHERECLAUSE

Parameters: String xWhere

Return type: Does not return a value

Description: Sets the SELF.WhereClause property to the value passed in xWhere.

SORTQUEUE
VIRTUAL

Parameters: Byte xOrder
 Byte xCol

Return type: Does not return a value

Description: Virtual method. When the queue is fully loaded with the entire data set or query set, this method takes over queue sorting, relieving the DB from having to sort the data. This happens with small data sets or when the operator browses the entire file. Or when the SELF.LoadType flag is set to HPROP:FullLoad.

SWAPCOLUMNS
LONG, PROC

Parameters: Long xCol1
 Long xCol2

Return type: Long, Proc

Description: Temporarily buffers one of the columns (into column 999), replaces it with the "dropped" column and then puts the buffered one back into place where the dropped one started.

SWAPPROPERTY **PRIVATE**

Parameters: No parameters
Return type: Does not return a value
Description: Private method. Pushes the final set of format strings into the listbox with PROP:Format.

TAKEACCEPTED **LONG, PROC**

Parameters: Long xField
 Long xEvent
Return type: Long, Proc
Description: Called from the WindowManager.TakeAccepted method and processes the EVENT:Accepted on a number of controls.

TAKEFIELDEVENT **LONG, PROC**

Parameters: Long xField
 Long xEvent
Return type: Long, Proc
Description: Called from WindowManager.TakeFieldEvent, this method processes various events for the listbox and also for various of the controls related to the SQL browse class.

TEMPLATEAUTOINIT

Parameters: No parameters
Return type: Does not return a value
Description: A placeholder method into which the template writes initialization code for the current instantiation of the class. This is cleaner than using a long parameter list.

UPDATE **VIRTUAL**

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Brings currently selected record from Q to BUFFER or DISK to BUFFER to Q depending on the setting of SELF.FetchRecordOnSelect True/False (See FetchRecord)

UPDATEFORMAT **LONG, PROC, VIRTUAL**

Parameters: No parameters

Return type: Long, Proc
Description: Virtual method. Saves the current browse format strings to the INI to be able to restore the browse in the state that the user left it. The SaveFormat property must be set to True (Default).

UPDATEHEADERS

Parameters: No parameters
Return type: Does not return a value
Description: Updates the Header queue with information about each header as it is in the listbox at the moment this method is called. If you need to change a listbox header at runtime, call this method after you are done updating the headers. That will ensure that the header doesn't revert back when the sort order is changed.

UPDATEQUEUE

VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Placeholder method only.

UPDATETOOLBARBUTTONS

VIRTUAL

Parameters: No parameters
Return type: Does not return a value
Description: Virtual method. Clone of the ABC version of the same name. Used to hook the toolbar into the browse.

CCSButtons Class

Class Properties

AFTERINSERTACTION	BYTE(1)
Description:	Flag HPROP:LocateAfterInsert, HPROP:RefreshAfterInsert.
ALERTKeys	&CCSKEYQ
Description:	A reference to the CCSKeyQ queue. The CCSKeyQ is declared as: <pre> CCSKeyQ Queue CCSKeyCode Long CCSKeyAction Long End </pre>
CANCTL	LONG (0)
Description:	FEQ of the Cancel control.
CANFLAG	LONG (2)
Description:	Cancel button accepted return flag (given to GlobReq).
CANKEY	LONG (001BH)
Description:	Cancel hot key.
CHGCTL	LONG (0)
Description:	FEQ of the Change control.
CHGKEY	LONG (000DH)
Description:	Change hot key.
CLSCTL	LONG (0)
Description:	FEQ of the Close control.
CLSFLAG	LONG (FALSE)
Description:	Close button accepted return flag (given to GlobalReq).
CLSKEY	LONG (001BH)
Description:	Close hot key.

DELCTL	LONG (0)
Description:	FEQ of the Delete control.
DELKEY	LONG (002EH)
Description:	Delete hot key.
GLOBREQ	&BYTE,THREAD
Description:	Reference to global request variable.
HIDeselect	BYTE(0)
Description:	True or false flag to indicate if Select button should be hidden.
INSCtl	LONG (0)
Description:	FEQ of the Insert control.
INSKEY	LONG (002DH)
Description:	Insert hot key.
KEYSALERTED	BYTE(FALSE)
Description:	True once the keys are alerted, false after Kill.
LIST	LONG (0)
Description:	List control FEQ.
POPUP	&POPUPCLASS
Description:	A reference to the ABC PopupClass.
SELCTL	LONG (0)
Description:	FEQ of the Select control.
SELFLAG	LONG (TRUE)
Description:	Select button accepted return flag (given to GlobReq).
SELKEY	LONG(0453H)
Description:	Select hot key.

USEPOPUPMENU
BYTE(FALSE)

Description: Flag that determines if the listbox will use the popup menu.

Class Methods

INITCANCEL
VIRTUAL

Parameters:

Long	xCan
Long	xCanKey
Byte	xRtnFlag
*Byte	xGlbReq

Return type: Does not return a value

Description: Virtual method. Initializes the Cancel button.

INITCLOSE
VIRTUAL

Parameters:

Long	xCls
Long	xClsKey
Byte	xRtnFlag
*Byte	xGlbReq

Return type: Does not return a value

Description: Virtual method. Initializes the Close button.

INITPOPUP

Parameters: No parameters

Return type: Does not return a value

Description: Initializes the Popup menu. Creates, instanciates and initializes the PopupClass instance. It also alerts the RightMouseUp key.

INITSELECT
VIRTUAL

Parameters:

Long	xSel
Long	xSelKey
Byte	xRequest
Byte	xRtnFlag
*Byte	xGlbReq

Return type: Does not return a value

Description: Virtual method. Initializes the Select button.

INITUPDATE
VIRTUAL

Parameters:

<Long	xIns>
<Long	xChg>
<Long	xDel>

Long xInsKey
 Long xChgKey
 Long xDelKey

Return type: Does not return a value
Description: Virtual method. Sets up the update buttons, Insert, Change and Delete, alerts hotkeys, sets up popup menu etc.

KILLKEYS

Parameters: No parameters
Return type: Does not return a value
Description: Disposes of the AlertKeys queue.

KILLPOPUP

Parameters: No parameters
Return type: Does not return a value
Description: Disposes of the Popup class.

REGISTERKEY

Parameters: Long pKeyCode
 Long pAction
Return type: Does not return a value
Description: Registers the key to perform the specified action, which can be InsertRecord, ChangeRecord, DeleteRecord or SelectRecord. This enables multiple hotkeys for each action.

TAKEALERTKEY

LONG

Parameters: Long pKeyCode
Return type: Long
Description: The keycode being pressed is passed to this function, and based on the keycode it will return the action keycode, i.e. Self.InsKey, Self.ChgKey or Self.SelKey. This enables multiple hotkeys for each action.

WDEBUG

Parameters: String pS
Return type: Does not return a value
Description: This method is implemented in the Locator class and the Buttons class. It uses the OutputDebugString api call to write to debug output port. This is best used in cooperation with tools such as DebugView from System Internals (www.systeminternals.com) By making this a public method, all our users can use it for their debugging purposes in browse procedures that use the SQL templates/classes.

CCSSizes Class

Class Methods

COLUMNWIDTH**LONG, VIRTUAL**

Parameters: STRING xText
 LONG xCtl
 LONG xWidth

Return type: Long

Description: Virtual method. This method returns the width that is needed for the column to be able to show the full column header text. Care must be taken not to call this method too early or in modal events as it creates a string control to measure the width of the header text.

CCSLocManager Class

Class Properties

OQ	&LocOwnerQ
-----------	-----------------------

Description: Queue of references to browse objects sharing locator. The LocOwnerQ is declared as:

```

LocOwnerQ  QUEUE,TYPE
BrwObj     &CCSSql1
BrwCtl     LONG(0)
LocCtl     LONG(0)
END

```

Class Methods

ADDLOCATOROWNER	VIRTUAL
------------------------	----------------

Parameters: CCSSql1 xBrwObj
LONG xBrwCtl
LONG xLocCtl

Return type: Does not return a value

Description: Virtual method. Adds another potential locator owner to the list of locator owners.

CLEARLOCATORS	BYTE, PROC, VIRTUAL
----------------------	----------------------------

Parameters: No parameters

Return type: BYTE,PROC

Description: Virtual method. Clear any outstanding locators on all browses in a shared locator situation. When no shared locators, clear just the locator for the relevant browse.

CLEARLOCATOR	BYTE, PROC, VIRTUAL
---------------------	----------------------------

Parameters: No parameters

Return type: BYTE,PROC

Description: Virtual method. Call the clear locator function of the browse object currently owning the locator.

CLEARLOCATORNOFORCE	VIRTUAL
----------------------------	----------------

Parameters: No parameters

Return type: Does not return a value

Description: Virtual method. Call the clear locator (no force) function of the browse object currently owning the locator.

CONSTRUCT

Parameters: No parameters
Return type: Does not return a value
Description: Constructor for the class.

DESTRUCT

Parameters: No parameters
Return type: Does not return a value
Description: Destructor for the class.

LOCATE

BYTE, PROC, VIRTUAL

Parameters: No parameters
Return type: BYTE,PROC
Description: Virtual method. Call the locate function of the browse object currently owning the locator.

SWAPLOCATOR

VIRTUAL

Parameters: LONG xBrwCtl
Return type: Does not return a value
Description: Virtual method. Switches ownership of the locator to the browse with FEQ xBrwCtl.

WDEBUG

Parameters: String pS
Return type: Does not return a value
Description: This method is implemented in the Locator class and the Buttons class. It uses the OutputDebugString api call to write to debug output port. This is best used in cooperation with tools such as DebugView from System Internals (www.systeminternals.com) By making this a public method, all our users can use it for their debugging purposes in browse procedures that use the SQL templates/classes.

CCSToolbarListboxClass Class

Duplicate of ABC ToolBarListBoxClass except that it does not use a reference to the ABC browse class. Which isn't in use here. The SELF.Browse property in this class is the list box FEQ. In fact, all the original class was doing is determining the list box FEQ by inspecting the SELF.Browse.ListControl property. This class is not exposed in the normal window template interface precluding any overriding through embedding.

Class Properties

BROWSE	LONG (0)
---------------	-----------------

Description:	List box FEQ. In the original ABC version, this is a reference to the ABC Browse Class.
---------------------	-----------------------------------------------------------------------------------------

Class Methods

DISPLAYBUTTONS	VIRTUAL
-----------------------	----------------

Parameters:	No parameters
Return type:	Does not return a value
Description:	Calls the ABC ToolbarTarget.DisplayButtons.

TAKETOOLBAR	VIRTUAL
--------------------	----------------

Parameters:	No parameters
Return type:	Does not return a value
Description:	

TAKEEVENT	VIRTUAL
------------------	----------------

Parameters:	< *LONG VCR > WindowManager WM
Return type:	Does not return a value
Description:	

TRYTAKETOOLBAR	VIRTUAL
-----------------------	----------------

Parameters:	BYTE
Return type:	Does not return a value
Description:	Virtual method. Calls TakeToolbar if the listbox is visible.

Compatibility and Technical issues

Currently we are not aware of any compatibility or technical issues with this product. We have done some testing on the SQL templates under Clarion 6 and they seem to operate without problems in Clarion 6, Early Access release 5 (EA5) as well as in Early Access release 4 - 4.5. If you experience problems with the SQL templates under Clarion 6, then please let us know as soon as possible either by email to support@icetips.com or by using the Icetips Software Online Support - see page 90 for more information about support.

Technical Support

We offer technical support by email, by newsgroup, or by an internet bulletin board. Starting in July 2003 we are also introducing a new and revolutionary support center which we call Icetips Software Online Support or I-SOS for short.

Email

Please email your questions to either support@icetips.com or wizard@icetips.com and we will get back to you as soon as possible. We usually respond to technical support emails within an hour.

Newsgroups

You can also post questions on the **Topspeed.Topic.Third_Party** newsgroup on the news.softvelocity.com news server or comp.lang.clarion, which you can get to at that same news server or on the web at:

<http://groups.google.com/groups?hl=en&group=comp.lang.clarion>

Internet Bulletin Board

We have a Internet Bulletin Board at the Icetips website, where you are welcome to post questions. We monitor it regularly, and there are quite a few people who visit it frequently. Go to:

<http://www.icetips.com/bboard/index.php>

Icetips Software Online Support, I-SOS

We are moving all of our software products to use our Online Support center, that we call I-SOS. This is available directly from within the products, from templates, compiled programs and anywhere where we feel that our users may need help. We are still in the development phase of this new technology, but expect to have it implemented in all our products by the end of 2003, probably sooner. The Icetips Cowboy SQL templates, version 6.0 is the first of our products to implement this in the templates. To log into our Online Support Center, please go to:

[Http://www.icetips.com/supportcenter/index.php](http://www.icetips.com/supportcenter/index.php)

Installed files

Following is a complete list of the installed files. Please note that it is possible that the dates/times and file sizes does not match completely with what is installed as this list was created before everything was completed. Also note that the file dates are in dd.mm.yyyy format.

Files in: 3rdParty\Docs\ITCowboySQL

Date	Time	Size	Filename
07/01/2003	04:23 PM	1,165,842	IcetipsCowboySQL.pdf
06/24/2003	09:13 PM	13,623	Versions.txt

(Note that these two files are probably not listed here with the right size)

Files in: 3rdParty\Images

Date	Time	Size	Filename
03/02/2003	03:39 PM	2,847	ITLogo.gif

Files in: 3rdParty\Template

Date	Time	Size	Filename
03/02/2003	03:39 PM	2,847	ITLogo.gif
07/02/2003	10:59 AM	205,327	Ccsabc.tpl
07/01/2003	06:02 PM	5,704	ITSQL60ABC.tpw
06/28/2003	04:54 PM	23,323	CCSABCEX.TPW

Files in: 3rdParty\Tools\ITInstall

Date	Time	Size	Filename
06/17/2003	10:55 AM	781,312	itutil.exe
07/01/2003	07:19 PM	1,409	itccs.itc

Files in: Bin

Date	Time	Size	Filename
06/30/2003	01:11 PM	19,968	itccs60.dll
06/30/2003	01:05 PM	18,432	itccs55.dll

Files in: LibSrc

Date	Time	Size	Filename
07/01/2003	05:00 PM	7,877	ccsbutns.clw
07/01/2003	05:00 PM	2,517	ccstoolb.clw
07/01/2003	05:00 PM	4,811	ccslocat.clw
07/01/2003	05:00 PM	147,424	ccssql1.clw
07/01/2003	05:00 PM	1,120	ccssizes.clw
07/01/2003	05:00 PM	1,320	ccslocat.inc
07/01/2003	05:00 PM	26,352	ccssql1.inc
07/01/2003	05:00 PM	3,616	ccsbutns.inc
07/01/2003	05:00 PM	941	ccstoolb.inc
07/01/2003	05:00 PM	815	ccssizes.inc

Last minute changes

Here are listed some last minute changes that were implemented too late for us to add them in the right places in the manual.

Default sort order

This is a new button on the "Sort" tab on the browse template. It allows setting the default sort order to either ascending or descending. Default is Ascending. Note that this only applies to the column that is active sort order column when the browse starts. So if you set a default sort column and Descending sort order, the browse will open with that column as active browse column in descending order.

Sort order

This button has been renamed to Sort order column so it is not confused with the new Default sort order.

Limitations

1. We have discovered that using Force INNER joins with listbox that has colors, icons or styles will mess up the listbox. We will have this fixed for next build which should be out by the end of July, 2003.

Changes from previous version

The following changes were noted down during the development phase of the Icetips Cowboy SQL version 6.000.

Version 6.000 Final release

June-17-2003 - June-30-2003

36. Documents Writing up documentation for the SQL templates.

June-16-2003

35. Template Cleaning up.

34. Template Buttons to go to our website and email us directly from the template.

June-14-2003

33. Template Implemented "Restore Child After Cancel-CCS Browse" extension for forms that also have CCS browses.

32. Template Cleaning up template.

June-13-2003

31. Template Cleaning up and going through template windows/prompts to prepare screenshooting

30. Template Implemented changes for ClarioNet compatibility originally done by Robert Rodgers.

29. Classes Swap Columns did not work. Fixed.

June-12-2003

28. Classes Default sort column added. Overrides saved browse format.

27. Classes Locator on dates checked. Locator on numbers used Like instead of = Fixed.

26. Classes Locator string with single quote in it (like Sam's) resulted in invalid SQL. Fixed.

25. Classes Work on fixing Swap Columns.

June-10-2003

25. Classes ActiveInvisible broken. Fixed.

24. Template UpdateButtons template: Procedure dropdown was too narrow. Fixed.

23. Template Work on implementing Windows Style tagging.

June-04-2003 - June-10-2003

22. Template Work on implementing better ways to use variables in the SQL browses.

June-03-2003

- 21. Template Implemented Reset Fields for item 20.
- 20. Classes Implemented FieldPairClass to allow reset variables that force the browse to reset. Hooked in through the ResetRequired method of the WindowComponent Interface.
- 19. Classes Testing WindowComponent - seems to work now without problems. Need to test in a bigger app.
- 18. Classes Replaced strings with CCSSQL1 with Self.ObjectName in debugging code.
- 17. Template Passes the %ObjList to the Init Method, see item 16.
- 16. Classes Init method now takes the name of the object from the template.
- 15. Template Implemented call to Updateheader in WindowManager.Open method call.
- 14. Template Kill method was not being called from templates resulting in memory leak. Fixed. Not true: It was being called from the WindowComponent Interface Kill method.
- 13. Template Implemented settings for sort order indicators and sort column indicators, see item 10.

June-02-2003

- 12. Template Modified the order in which the window property is set, the Init and TemplateAutoInit are called. Added a priority space between the call to Init and TemplateAutoInit so code can be placed there, for example to change the sort order indicators etc.
- 11. Classes Changed the way the column headers are built up to accommodate for item 8 and also to make it possible to use runtime translation. Previously the column headers would not register correctly (i.e. they would write to the trn as "<+header>" for the active sort header instead of just "header". The translation of the column headers would also not stick because the translated headers weren't in the column queue in the SQL class. Fixed.
- 10. Classes Implemented methods and properties to change the sort order indicators (+/-) and sort column indicators (</>).

May-30-2003

- 9. Classes Added properties and methods to change the order signs. Default values set in Init method.
- 8. Classes Reverted the Reset things. Don't think there is any need for them now with the WC working so well!
- 7. Classes Tested WindowComponent implementation. Needs some more work.

May-27-2003

- 6. Classes Implemented Reset control registration and event registration. Controls can be registered along with events
- 5. Template Implemented ActiveInvisible in templates. In testing.
- 4. Classes Implemented registration of WC interface. In testing.
- 3. Template Classname appears on the Action tab now for convenience.

May-11-2003

- 2. Template Code to extract keycodes for update buttons, would be out of scope and not execute, leaving the impression that the update button keys had not been properly selected. Fixed.
- 1. Classes Implemented ActiveInvisible. It basically shortstops the ResetQueue method if it is set to false and the control is not visible. Basically duplicated methods from ABC. Need to duplicate methods to trigger changes and implement in templates. No testing yet.

Version 6.000 Beta B, May 5, 2003

KNOWN ISSUES in 6.0, Beta B

- 1. Template Child sync on multi-component keys only works correctly with ONE child browse. See item 14 in Beta A. Will be fixed for next release.
- 2. Documents Still not there. It's on my (already heavy) schedule for May 19 - May 25.

Changes from Beta A

- 1. Template Added conditional compile defines for ClarioNet compatibility.
- 2. Template Added option to put literal filters into the filter field. If the filter starts with an exclamation mark (!) the filter is not quoted by the templates and it's up to the developer to supply the exact clarion/SQL code that goes into the filter.
- 3. Template Added several new options to the Update button template. Suppress clear and Clear record are now available as checkboxes. This makes it easy to prime the record in the browse. Also added a Pre and Post priming. When creating a record RI must be maintained and certain required RI fields may need to be populated. This is done with the Pre-priming. Once the record is actually created, these fields can be reset to NULL or set to another value in the Post priming which is done before the record is sent to the update form. The assignment can be done to a value, variable, NULL or by calling a procedure to fill the variable. This can be local variables or table columns. A button was added to jump stright into the the PrimeRecord embed from Insert Options to make it easy to add any additional code.
- 4. Template ~~Modified the CONTROLS part of the SQLBrowseNL template so that it pops up the listbox formatter when the table has been dropped. ,FROM(SQL:Q) added to the LIST control. (ROLLED BACK - this generates a FROMNo parameters queue name that is not compatible with the generated queue label. This would have to be fixed in several places in the templates and I'm not going to do that right now)~~
- 5. Template Added support for Popup menu. Added mimic support for Select, Insert, Change and Delete buttons. Implemented in CCSButns.inc, CCSButns.clw, CCSSQL1.clw and CCSABC.TPL
- 6. Classes Implemented options to add alerted keys to insert/change/delete. Seems to work fine. Changes in CCSButns.inc, CCSButns.clw and CCSSQL1.clw.
- 7. Classes Prototyped OutputDebugString in CCSButns.clw to make it easier to debug the classes. Method wDebug is available for anyone to use.

8. Classes The CCSSql1 class now implements the WindowComponent Interface for future use. So far, responses to Kill, TakeEvent, Reset and ResetRequired interface methods has been added.
9. Classes Init and Kill methods added to class. Kill method calls KillPopup and KillKeys methods and removes the WindowComponent interface.
10. Template All appropriate prompts changed from @S... to EXPR
12. Template Added support for additional hotkeys as pr. changes in item 6 to the classfiles.
13. Template Re-instated old locator and renamed the Icetips locator to ITSQlBrowseLocator. That way it will not break any functionality in existing CCS applications.
14. Template Moved prepare code around in the SQLBrowseUpdateButtons template to eliminate "undefined symbol" error on the procedure information. Think I've got this issue settled.
15. Template Copyright changed to Icetips Software and Icetips Logo added to templates.
16. Classes Copyright changed to Icetips Software.

Version 6.000 Beta A, February 20, 2003

Changes from 5.5

1. Template Modified the Locator control template so that it uses the default font and also so that it prompts for the string, entry and button, not string, button and entry. Also modified the entry field to be 10 du high instead of 12 du.
2. Template Modified the templates to make an option to force use of files if LazyOpen is on.
3. Template Fixed locator so now there can be multiple locators on a window.
4. Template Icon selection now has a file selection dialog box and conditional icons have file selection dialog and an Expression field to select condition.
5. Template Added an option to synchronize child browses with relations in compound keys - enables multiple field assignment for synchronized filtering.
6. Template Full support for Color added.
7. Template Full support for conditional Colors added.
8. Template Full support for Icons added.
9. Template Full support for conditional Icons added.
10. Template Full support for Styles added.
11. Template Full support for conditional Styles added.
12. Template Style Builder. Lets you build styles and name them and you use the name as Styles instead of numbers like ABC does.
13. Template Full greenbar support - uses Steve B. Greenbar fully integrated into the SQL templates (no extension, just a button on the SQL template window)
14. Template Supports child synchronizing on compound keys (original only supported one field, which is not going to work on tables that are set up properly for replication for example)
15. Classes Support for SuppressClear in PrimeRecord. Allows multi-component keys to be autoincremented properly and fields to be populated when Clarion handles autoinc.