

BoxSoft
Corporation

Super Browse - Documentation

Copyright © 1989-2014, BoxSoft Corporation, All Rights Reserved.

Table of Contents

Foreword	0
Part I Getting Started	3
1 Introduction.....	3
2 RTFM Warning!!!.....	6
3 Installation.....	7
4 Upgrading from Earlier Versions.....	11
Part II Template Usage	12
1 Adding SuperBrowse to your Applications.....	12
2 Browse Entry (Edit-In-Place).....	13
3 Restore Child After Cancel.....	24
4 Conditional Formats.....	26
5 Tab Pop-ups.....	28
6 Select Mode Cancel Button.....	29
7 Action Headers.....	31
8 Locator Tabs.....	35
9 Color List Selector.....	37
10 Bold Selected Tab.....	39
11 Display Locator.....	41
12 Global LIST Line Height.....	42
Part III Appendices	43
1 Example Programs.....	43
2 Project Defines.....	44
3 Troubleshooting.....	46
4 Contacting Technical Support.....	48
5 License Agreement.....	49
Index	50

1 Getting Started

1.1 Introduction

The SuperBrowse templates include a variety of features for you to empower and embellish your Browse windows. All features have been developed with the following goals:

1. The feature must be easy to implement.
2. It must work with the regular Clarion templates.
3. All code must be visible in the template (i.e. no black-box DLLs).
4. The resulting programs should work intuitively and quickly.

Although we have a number of templates included here, the shining star is definitely our implementation of Edit In Place (EIP). We're taken what Clarion provided in ABC to a whole new level. It's easier to implement, more flexible, and far more powerful. Of course, we're never finishing making improvements. If you have ideas, suggestions or comments (either positive or negative), please pass them along. They're greatly appreciated.

Browse Entry - Many developers and users dislike the Browse+Form metaphor. They would prefer to enter data directly on the browse. Clarion ABC has provided basic Edit-In-Place (EIP), but everyone has admitted that it is only a starting point. Well, we've taken the existing platform, and expanded to the point where you'll be very happy with the results.

Restore Child After Cancel - Whenever you allow a user to change child files on a parent's update form, there's a good chance that the user will cancel the operation without realizing that the changes to the child records are not cancelled as well. This template restores the child file(s) to their original state. (See "SuperBrowse versus SuperInvoice" below for more information.)

Conditional Formats - This combination of an extension and child control templates lets you provide your users with a different display format, depending on the specified conditions. You can make a special format for each tab on your browse, or use any other conditions as you see fit.

Tab Pop-ups - This global extension template adds pop-ups to all of the sheets in your application (for Browsers, Forms, nested sheets, etc.). When the user right-clicks on the tab region, they see a menu of all the available tabs with a checkmark beside the current tab. This is very handy if you like to add many tabs to your sheets: It's much faster than using the sheet's scroll buttons, yet it allows you to keep your tabs on a single row.

Select Mode Cancel Button - This global extension template automatically changes your [Close] button to [Cancel] whenever a browse is called in SelectRecord mode. This is far more intuitive for your users, and saves you the hassle of explaining that "Close" sometimes means "Cancel".

Action Headers - The Windows 95 explorer and many other database programs have popularized the ability to press the header of a list box to sort the list ordered by that column. This extension template allows you to achieve this feat. If you are using a custom sort order

and you press the header a second time, the order is reversed. You can use the headers to select your regular Tabs (to use the built-in keys), or you can use a custom order of one or more fields (using PROP:Order). You can also execute your own source code.

Locator Tabs - Many applications have "rolodex-style" locators for their browses (see the Solodex example that comes with Clarion.) Some users love use their mouse whenever possible, and they prefer this visual locator over having to type the locator value. This control template places a sheet of locator tabs on your window. You can place them above, below, to the left or right of your browse box. They automatically scroll if there is not enough space to show all letters. You can optionally support spaces and numbers in your locators. (Using Locator Tabs instead of Locator Buttons is an improvement, as you can easily change the position of a sheet, versus individually repositioning many buttons.)

Color List Selector - Another confusing issue for users involves the list selector bar. Regardless of whether your list box has focus, the highlighter bar is very bold. It is difficult to see the insignificant dotted line indicating which control actually has focus. The problem is increased if you have multiple list boxes on the same window. This extension template changes the selector bar color when the list box does not have focus, making the current control much more obvious. You can use this feature on any list box (not just Clarion's BrowseBox). The global template allows you to apply this improvement throughout your APP, and you can tweak it where necessary with a local extension template.

Bold Selected Tab - When we watch users, we often notice that they are confused as to their current tab / sort order. They will often change tabs, then change back again, just to be sure that they are in the right place. This extension template will highlight the currently selected tab using the bold attribute, text color and/or tab color. (You can use this for non-browse sheets too.) Again, there is a global template to implement this feature across your entire application, with local extension override.

Display Locator - This control template uses a string on the window to display the current value of the incremental locator. This enables your users to see the locator value as they type it. This is superior to placing the actual key component on the window, as the DisplayLocator changes it's use variable based upon the Browse conditions.

SuperBrowse versus SuperInvoice

Many of you may wonder about the difference between "SuperBrowse" and "SuperInvoice". The SuperInvoice template is designed to support "in-place" data-entry on a list box. The entire transaction (the parent and all children) is held in memory until the [OK] button is pressed. With SuperBrowse, these changes are saved to the disk as they are entered.

Even with the "Restore Children after Cancel" template, it is still better to use SuperInvoice when possible. There are two situations where you will be forced to use SuperBrowse:

1. If there are too many records in the child to load into memory.
2. If you have multiple child files that you wish to edit (either multiple children for the same parent, or children and grandchildren).

ABC and Legacy Template Chains

This documentation pertains to both the ABC and Legacy (a.k.a. "Clarion") Super Template sets. In some situations we've implemented features in ABC that are not in Legacy, primarily because the old template chain was to be phased out. Due to customer pressures, however, Soft Velocity decided to reinstate support for the Legacy/Clarion chain.

Some of the Super Template features that are only in the ABC chain would be very difficult to implement in the legacy chain. However, we'll attempt to do this wherever it seems feasible to us. We apologize if this causes you any inconvenience. Please feel free to contact us if there's a particular feature in ABC that you would like to see in the Legacy chain, and we'll see if your needs can be accommodated.

For more information on the SuperBrowse templates, see the following topics:

Adding SuperBrowse to your Application

- Browse Entry
- Restore Child After Cancel
- Conditional Formats
- Tab Pop-ups
- Select Mode Cancel Button
- Action Headers
- Locator Tabs
- Color List Selector
- Bold Selected Tab
- Display Locator
- Global LIST Line Height

1.2 RTFM Warning!!!

It is very important that you read this documentation. If you follow the instructions step-by-step, then the usage is very simple. It is almost *impossible* if you try to do it on your own!

1.3 Installation

Installation Directory Structure

NOTE: As of version 6.6, we've changed our installation to the defacto "3rdParty" directory structure. (In Clarion 7 this is actually the "Accessory" directory.) Your old CLARIONx\SUPER directory has been renamed to CLARIONx\SUPER-OLD.

Once you've finished running the installation program, you should see the following structure under your C55 or Clarion6 directory:

```
C:\CLARION6, C:\C55, etc.
+-LibSrc      STA*.INC      (ABC headers)
+-3rdParty
| +-Bin ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
| +-Template  STA?*.TPL, STA*.TPW      (ABC chain)
| |          STC?*.TPL, STC*.TPW      (Clarion chain)
| +-LibSrc    STA*.INC, STA*.CLW, STA*.TRN (ABC chain)
| |          STC*.INC, STC*.CLW, STC*.TRN (Clarion chain)
+-Images
| `--Super   *.ICO, *.CUR, *.WMF, *.GIF
+-Docs
| `--Super   *.PDF      (Documentation)
`--Vendor
   `--Super
      +-QBE
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      +-Tagging
      | +-Examples
      | | +-ABC *.DCT, *.APP, *.TPS (Examples) (ABC chain)
      | | `--Clarion *.DCT, *.APP, *.TPS (Examples) (Clarion chain)
      | `--Source
      | | +-ABC *.TXD, *.TXA, *.DCT (Source) (ABC chain)
      | | `--Clarion *.TXD, *.TXA, *.DCT (Source) (Clarion chain)
      `--Etc.
      +- . . .
`--SUPER-OLD      (ABC headers)
   `-- . . .
```

For Clarion 7 and later versions it should look like this (note the two trees):

```
C:\Program Files\SoftVelocity\Clarion 7
`--Accessory
   +-Bin          ST_*.HLP, ST_*.CNT, STAB_CNV.DLL
   +-Template
   | `--Win       STA?*.TPL, STA*.TPW      (ABC chain)
   |              STC?*.TPL, STC*.TPW      (Clarion chain)
   |              STMH*.TPW                (Shared)
   +-LibSrc
   | `--Win       STA*.INC, STA*.CLW, STA*.TRN (ABC chain)
   |              STC*.INC, STC*.CLW, STC*.TRN (Clarion chain)
   +-Images
   | `--Super     *.ICO, *.CUR, *.WMF, *.GIF
   `--Docs
      `--Super     *.PDF      (Documentation)
```

```

"My Documents" or "Shared Data" (depending on OS)
^-Clarion 7\Accessory
  ^-Super
    +-QBE
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    +-Tagging
      | +-Examples
      | | +-ABC          *.DCT, *.APP, *.TPS (Examples)      (ABC chain)
      | | ^-Clarion     *.DCT, *.APP, *.TPS (Examples)      (Clarion chain)
      | | ^-Source
      | | +-ABC          *.TXD, *.TXA, *.DCT (Source)         (ABC chain)
      | | ^-Clarion     *.TXD, *.TXA, *.DCT (Source)         (Clarion chain)
    ^-Etc.
    +- . . .

```

To prevent conflicts between old Super Template files and same-named files in our new directory structure, the new installers attempt to delete the old files. If it encounters problems, then an error will be reported during the installation. Then you must delete any of the following files from the old directories, if they also exist in the new directory structure:

```

C:\CLARION6, C:\C55, etc.
+-LibSrc          STAB*.CLW, STCL*.CLW, STAM*.CLW, STCM*.CLW,
|                STAB*.TRN, STCL*.TRN, STAM*.TRN, STCM*.TRN,
|                STDEBUG.*
+-Template        STAB*.TP?, STCL*.TP?, STAM*.TPW, STCM*.TPW,
|                STGROUPS.TPW, STDEBUG.TPW
^-Bin             ST_*.HLP, ST_*.CNT, ST_*.GID

```

For example, you can use a tool like the indispensable Beyond Compare (www.scootersoftware.com) to investigate the contents of C:\Clarion6\LibSrc and C:\Clarion6\3rdParty\LibSrc. View only files matching the mask ST*.* and hide all "orphans", which will show the files that exist in both directories. Delete the files from C:\Clarion6\LibSrc, and then do the same for the Template and Bin directories.

Filenames and Product Abbreviations

- Super Template filenames generally start with the letters "ST". That's about all you can go on most of the time. (Our image files don't follow this convention, but they are sequestered in the Image\Super subdirectory.)
- The next two letters are usually AB (for the ABC chain) or CL (for the Clarion/legacy chain). One exception is Super Stuff (MH), which uses AM, CM and MH. Also, if both the ABC and Clarion chain share a TPW, then the AB/CL are skipped and it goes on to the product abbreviation. (Again, Super Stuff is an exception, as it uses MH for the shared files.)
- There are several TPWs that are shared by multiple Super Templates: STGROUPS.TPW, STABABC.TPW, STBLDEXP.TPW, STDEBUG.TPW
- The last four characters:
 - For TPL files, the last four letters are an underscore, followed by one of the following

suffices. The exceptions are STABAEQB.TPL and ST?M_STF.TPL.

- For TPW files, the last four letters may match one of these in its entirety, or be followed by additional characters denoting the special purpose files.
- Super Stuff (MH) is an exception, in that it uses STcMxxxx, where "c" is the chain of A or C, and "xxxx" denotes the special purpose.

AEQB	Super QuickBooks-Export (i.e. Accounting-Export QuickBooks)
BRW/BW	Super Browse
DIA	Super Dialer
FF	Super Field-Filler
IE	Super Import-Export
INV	Super Invoice
LIM	Super Limiter
PCD	Super Passcode
QBE	Super QBE
SEC	Super Security
TAG	Super Tagging
MH/STF	Super Stuff (MH) (a.k.a. <i>The "MikeHanson" Templates</i>)

Update the Redirection File

The installation program is able to update your redirection file automatically. If you decline the option during the installation, then you will have to edit the redirection file yourself. The three things that must be found are the Templates, LibSrc and Images. For example, you might make the following changes to the the *.* entry in Clarion 6:

```
*.* = .; %ROOT%\examples; %ROOT%\libsrc; %ROOT%\images; %ROOT%\template; %ROOT%\3rdParty\template; %ROOT%\3rdParty\libsrc; %ROOT%\3rdParty\images\super
```

In Clarion 7 and above it will be more like this:

```
*.* = %ROOT%\Accessory\images; %ROOT%\Accessory\resources; %ROOT%\Accessory\libsrc\win; %ROOT%\Accessory\template\win; %ROOT%\Accessory\images\Super
```

There are *.RED examples in the SUPER\DOC directory.

Register the Templates

Clarion allows you to have multiple template sets accessible in the same application. It does this with the Template Registry. To use a Super Template, you must register it first. The installation program attempts to do this for you, but in case it fails, or if your registry becomes corrupted, then you must register them manually.

1. Load Clarion, then select the "Setup / Template Registry" pulldown menu option.
2. Press the [Register] button.
3. Select C:\CLARION\3rdParty\Template\ST_*.TPL (ABC) or ST_*.TPL (Clarion). The directory name may not exactly match your system.

Assuming this all went without a hitch, you're ready to start using the templates.

1.4 Upgrading from Earlier Versions

There are two aspects of our templates that have changed from the prior version. In the case of BoldTabs and ColorSelector, it is now necessary to add a global extension template for each of these. The local template is only used to override the global settings, and is no longer explicitly required.

The other change is more significant. With the release of Clarion ABC, we've decided to tow the line and work with Clarion's form of Edit In Place. This makes some things easier and more powerful, and makes other things impossible. Therefore, edit in place will not work the same as it did before. Some of the key differences are:

1. You don't need to declare the associated edit controls for each column (or specify the control range for the template). Instead, you declare the settings for each individual column. If you wish, you can still create your own control for each column, or you can let the template create it for you, or use any combination of the two.
2. You are not always in "edit mode". This means that you scroll around like a normal browse box, then hit the [Change] button (or double-click) to go into edit mode. The old template did not include a Change button with the auto-populated controls. To add a change button, add a normal button, then edit the screen source and copy the #SEQ() attribute from the Insert or Delete button over to the new Change button.
3. Some of the navigation keys while in edit mode (like PgUp, PgDn, CtrlPgUp, etc.) are not supported any longer. This is because we don't have control of it. (It's handled in the Clarion BrowseClass.AskRecord method.) We've requested that SoftVelocity make it possible to add these with a future version of Clarion.
4. You have many new types of edit controls now, including ENTRY, SPIN, CHECK, LIST+DROP, Lookup Button, and FileDrop.
5. At this point, we've removed the support for a regular update form. We'll be adding it to a future version, as soon as we settle on the best way to do it.
6. Your EIP embed code will be much different. For example, instead of issuing a CYCLE to stay in the same field, you must return LEVEL:Notify from the TakeAccepted method.

2 Template Usage

2.1 Adding SuperBrowse to your Applications

Many of the SuperBrowse templates are style-related, and they are implemented as global extensions. If you need to override the global settings, there is usually a local extension template that will accommodate your needs. The local extension is useless until the global template is put into place. These types of templates include:

- Color List Selector
- Bold Selected Tab
- Tab Pop-ups
- Select Mode Cancel Button

There are also some control templates, which are populated onto the Window. Most of them require that a browse box already be present (in which case they will not appear in the list of available templates.) The control templates include:

- Browse Entry
- Locator Tabs
- Display Locator
- Conditional Formats

The rest of the templates are extension templates, and are added on the Extensions window. In some cases, they require a BrowseBox extension to be present already. If this is the case, then you must highlight the desired browse extension before hitting the [Insert] button.

- Restore Child After Cancel
- Action Headers
- Conditional Formats

You may notice that "Conditional Formats" was on both of the previous two lists. This is because there is a control template which is added repeatedly to define each of the formats, and an extension template to specify all of the conditions.

2.2 Browse Entry (Edit-In-Place)

(Procedure Control Template)

Many people hate the Browse+Form metaphor. They would prefer to enter data directly on the browse. Clarion offers basic Edit In Place (EIP) in the ABC templates, but most feel that is lacking in features and very difficult to implement in a "real world" situation. The BrowseEntry control template irons out the wrinkles, and gives you a much more usable EIP.

Populating the Template

This is a control template, so you must populate it onto your window. Go into the window formatter, and press "Populate / Control Template" from the pulldown window (or press the bottom right button on the populate toolbox).

The template requires Clarion's BrowseBox. If you don't already have one on your window, then the "BrowseEntry" template will not be available for selection. If you have only one BrowseBox, then it will automatically be selected. If, however, you have more than one BrowseBox on your window that does not yet have LocatorTabs associated with it, then you will be offered a list of available browse boxes with which you can associate this new control template.

You will see three buttons appear: [Insert], [Change] and [Delete]. You can change the text on these to anything you desire.

The [Insert] button adds a new record to the BrowseBox. If the resulting record belongs somewhere else in the browse sorting order, then it will be moved there after the user finishes editing the row and attempts to move to another row. All auto-numbering and field priming is performed by the ABC file classes.

The [Change] button also goes into EIP mode, but edits the currently highlighted record, rather than adding a new one. The user can also double-click on the row to invoke EIP.

The [Delete] button calls the standard `Relate:Primary.Delete()` function to perform the delete operation. This is essentially the same as the regular update form.

NOTE: If you try to use Clarion's regular update buttons alongside SuperBrowse's EIP, you will always get the form. This is due to the approach that the BrowseClass uses to handle update procedures. We will be improving this interoperability with an upcoming version.

Automatic Controls

The BrowseEntry template will create a variety of control types, depending on your dictionary settings. The most common type will be an ENTRY control. If you specify lookup parameters along with requesting a button, the button will also be provided. If your dictionary settings specify `must be true` or `must be false`, then a CHECK field will automatically be used. If you specify "Must be in list", it will use a drop list of choices. It also adds attributes like UPR and CAP.

Wherever possible, we've tried to provide intelligent automation. We're certain, however, that we haven't anticipated all possibilities. If the template doesn't implement the control and/or

options that you expected, please contact us so that we can make the template as intelligent as possible.

Also, you can create your own controls and tell the template to use your control for a particular column. (See the following section for more details.)

Populating Manual Controls

The templates will do its best to create an EIP control based upon your dictionary settings. If this is not appropriate, you have the option of creating your own entry controls to be used during EIP. This is beneficial in a number of situations:

1. You have a bunch of field settings (like justification, typing mode, colors, fonts, etc.). These are more easily done with the normal control editing windows.
2. You have a checkbox column, but you don't want your EIP checkbox to borrow the text from the column's header.
3. You want to use the FileDrop template to perform a lookup.

In these cases (and probably a few others), creating your own controls makes sense. Just create the control as you usually would, then you can tell the template to use that one instead of creating its own. Of course doing this takes extra time on your part, so don't do it if it isn't necessary.

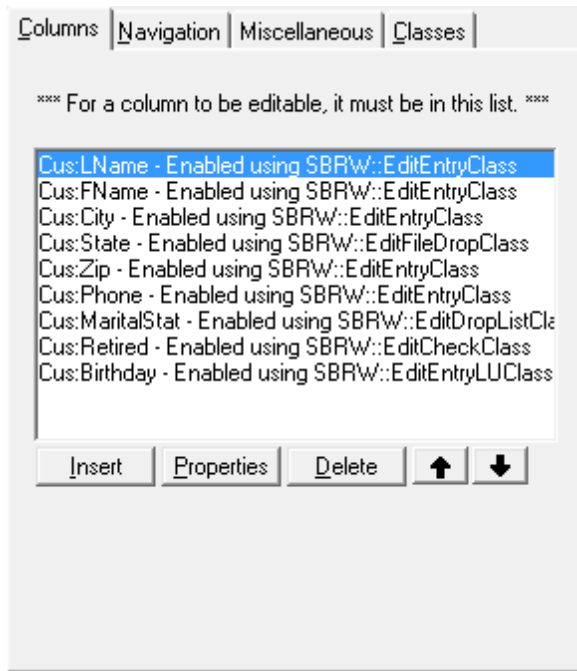
The types of controls allowed at this time are ENTRY, SPIN, CHECK, and FileDrop(ABC). The controls don't have to be full size (except, currently, for drop lists). They will automatically be expanded and placed onto the BrowseBox at run-time. Therefore, you can stuff them in a corner of the window. For an example of this, see SUPER\EXAMPLES\BROWSE\TEST.APP.

NOTE: If you are creating your own controls, be aware that any code you put into the regular event handling embeds will not be executed. There are special embeds for EIP event processing, and they are more easily accessible from the template prompts and the full procedure's embeds window.

Template Settings

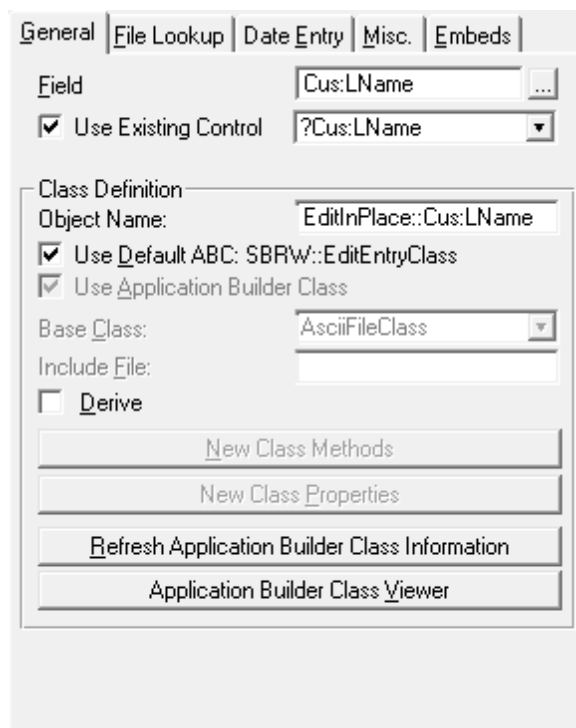
Once you finish populating your entry controls, you can go into the Actions of one of the Insert, Change or Delete button (or go to the Extensions window from the Procedure Properties). You'll find the following settings:

Column Settings



This lists columns that can be edited. If you wish to edit a column, you must include it in this list. Each entry has the following settings:

Column: "General" Tab



Field - This is the column from the list box, which you want to edit.

Use Existing Control - If you have created your own entry control, then turn this ON.

Entry Control - This is your window control that is assigned to this column. (This setting is only visible if the above setting is ON.)

Class Definition - These are the standard class settings that Clarion uses throughout it's ABC templates. If you are deleting and adding columns to the list, be careful that your Object Names are unique. It is strongly suggested that you do not change the change the default class unless you really know what you're doing!

Column: "File Lookup" Tab

The Lookup settings control whether your field is validated against a file. If you specify nothing here, but you have specified "must be in file" in the dictionary, then that will be done instead. If present, these settings override any lookup settings in the dictionary.

Lookup Key - This is the key in the related file to be used for the lookup.

Lookup Field - This is the field in the key. Normally it should be a single-field, primary key.

Lookup Procedure - This is optional. It names a procedure to be called in Select mode if the entered value does not exist in the related file.

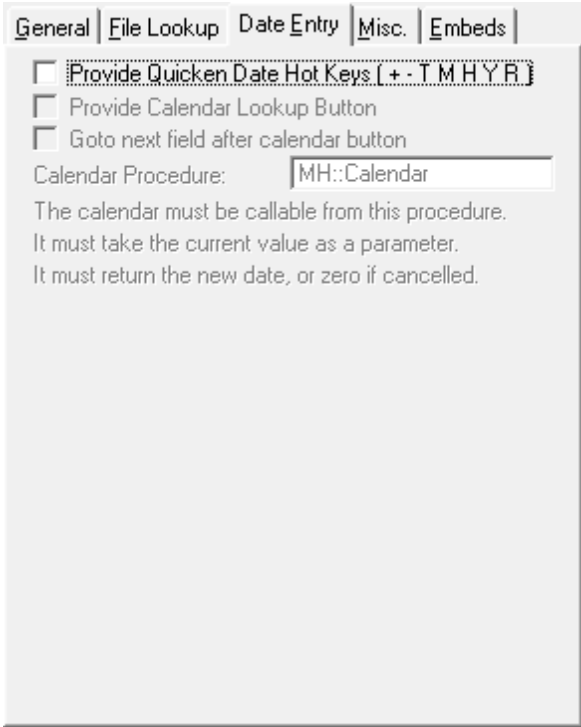
Lookup When - You can have the lookup performed when the field is selected, when it is accepted, or both. If there is no Lookup Procedure, then only Accepted is

performed.

Provide Lookup Button - If you have not specified your own entry control and you have specified a lookup procedure, then you can have the template automatically provide a lookup button, which the user can press to force the lookup procedure to be called. This button will use ST_LKUP.ICO (ellipsis).

Goto next field after lookup button - Do you want focus to pass to the next field after the user has selected an item with the lookup procedure?

Column: "Date Entry" Tab



General | File Lookup | **Date Entry** | Misc. | Embeds

Provide Quicken Date Hot Keys (+ - T M H Y R)

Provide Calendar Lookup Button

Goto next field after calendar button

Calendar Procedure:

The calendar must be callable from this procedure.
It must take the current value as a parameter.
It must return the new date, or zero if cancelled.

Provide Quicken Date Hot Keys - T for today, + for next day, - for previous day, M for start of month, H for end of month, Y for start of year, R for end of year, W for start of week, and K for end of week.

Provide Calendar Lookup Button - If you want your date fields to have a lookup button, which calls a procedure for a calendar window, then turn this ON.

Goto next field after calendar button - Do you want focus to pass to the next field after the user has selected an item with the calendar?

Calendar Procedure - This is the procedure that's called when the user hits the date field's lookup button. It must be callable from this procedure, it must take the current date value as a LONG, and must return the new date, or zero if cancelled.

As a default, it assumes MH::Calendar, which is included in CALENDAR.TXA found

in CLARION\SUPER\SRC_ABC\BROWSE. Just import it into your APP. (If you have a multi-APP system, then you probably will want to import it into the base APP, and then export it so it's available in the rest of your APPs.)

Column: "Misc." Tab

General | File Lookup | Date Entry | **Misc.** | Embeds

Place button beyond right border (if applicable)

Checkbox uses header text (if applicable)

"Disable Column Edit" Condition (True=Disabled)
Child fields must be prefixed with "BRW1.Q.", as in
"BRW1.Q.Cus:LName".

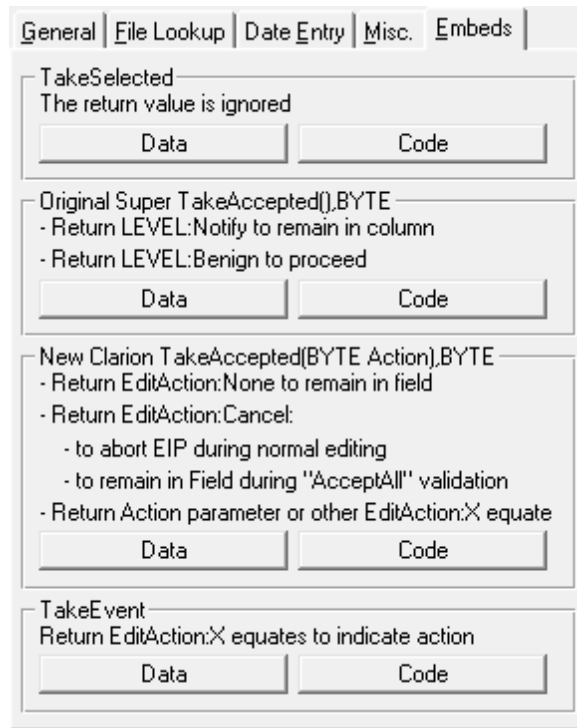
Hide column when condition is true
The hiding applies when the window opens, and NOT
dynamically to each row based upon its current
contents.

Place button beyond right border (if applicable) - If you are performing lookups, or you have a drop box or a spin control, then you will have a button as part of the control. This can take up valuable space within the column. If you can't spare this space, then turn this ON. The control will be expanded so that the button is beyond the right border of the column.

Checkbox uses header text (if applicable) - Do you want the checkbox control to copy the text from the column's header?

"Disable Column Edit" Condition - If there are situations where a particular column should not be edited, then enter that condition here. If the condition is TRUE, then the column will be skipped during EIP operations. Remember that your condition must use the "mirror" variables in the queue fields, and not the original fields themselves. For example, use BRW1.Q.Pre:MyField instead of just Pre:MyField (see "Methods and Embed Code" later in this section).

Column: "Embeds" Tab



These buttons let you quickly jump to the embeds associated with this column. In most cases you will be using TakeAccepted, occasionally TakeSelected, and rarely TakeEvent. There are other EIP methods that can be accessed from the main embeds window, but these buttons are handy for the most common needs.

For pointers on writing code for these methods, keep reading.

Navigation Tab

The screenshot shows the 'Navigation' tab of a settings dialog. It contains several sections with 'Save' options and checkboxes:

- Action upon tab at end of row:** Save is set to 'Always' (highlighted in blue). The 'Remain editing' checkbox is checked.
- Action upon ENTER key:** Save is set to 'Always'. The 'Remain editing' checkbox is checked.
- Default action upon arrow key:** Save is set to 'Always'. Both 'Remain editing' and 'Retain column' checkboxes are checked.
- Action upon focus loss:** Save is set to 'Always'.
- Insertion point:** Set to 'After'.
- Action on Delete:** Set to 'Always'.

At the bottom, there is a button labeled 'Field Priming on Insert'.

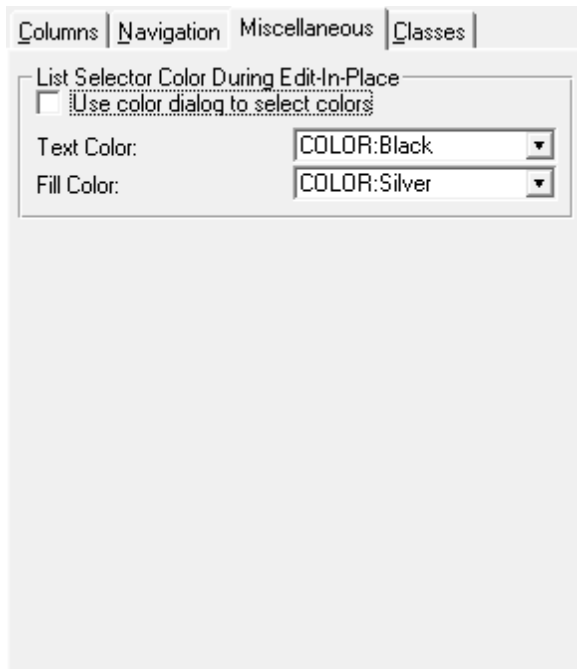
Each group has a Save setting, which determines whether the record is saved after the various operations. The options are Default, Always, Never, and Prompted. We suggest that you usually use "Always", unless you have a good reason to the contrary.

There are also a number of Remain Editing options. These are up to your personal preferences.

When pressing the Up and Down arrow keys, you have the choice to Retain Column, or to go to the first editable column on the row.

You'll also find options to determine how deletes are intervened, as well as field priming settings for inserts.

Miscellaneous Tab



List Selector Color During Edit-In-Place - The bold highlighting on the row can be distracting to the user during EIP. To change the selector bar color during EIP, use these settings. It makes it much easier to see which field they are in.

Methods and Embed Code

When it comes to hand-code, there are two very important things to realize about the new Edit-In-Place:

1. The controls are not updating the file's fields. Rather, the browse's queue buffer receives all field values until the row is saved. Therefore, it is useless to say:

```
Itm:Extended = Itm:Price * Itm:Qty
```

Instead, you should say:

```
BRW1.Q.Itm:Extended = BRW1.Q.Itm:Price * BRW1.Q.Itm:Qty
```

If you don't know the name of your Browse, look in the Classes tab of your Browse's extension settings. The object name is there. (You can also change it there.)

This makes it difficult for normal templates, as they expect to work with regular fields. We made our template explicitly handle Clarion's FileDrop template, because we felt it was a necessary tool, and we later supported FileDropCombo's too. Supporting each additional template will required special treatment.

2. EIP is controlled by the AskRecord method of the BrowseClass. It uses its own ACCEPT loop, and it passes very few events through to your own code. We've tricked it into

telling us when it's entering and exiting a field, but the EVENT() will not be EVENT: Selected and EVENT:Accepted.

Normally you should use the TakeSelected and TakeAccepted methods to handle your validation code. When you want to stay in a field (e.g. because of validation failure), then you must return Level:Notify from TakeAccepted. For example:

```
IF BRW1.Q.Itm:Type = 'X' !If it's a bad value
  RETURN Level:Notify    !Stay in the same field
END
```

If, instead, you're using TakeEvent, then you must return EditAction:None to stay in the same field.

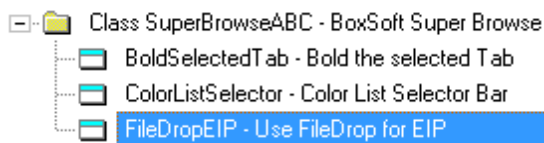
The easiest way to access these embeds is through the individual column settings in the extension. You can also get their through the regular Procedure Embeds button.

Adding the Regular Controls for "In-Place" Entry

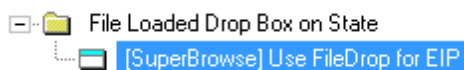
If you're using the ABC chain, then the template will automatically create controls for data entry at runtime. There are a few situations where you might want to create your own entry controls, though. For example, you may want to use the FileDrop or FileDropCombo. Or you may want to use a checkbox, but you want to use a different description from the automatic usage of the column header. Or you may want to use a picture that's different from the one defined in the dictionary. Or you might want to use a special font.

For some of these (like the picture, color, checkbox text, font, etc.), you can use property syntax assignments in the CreateControl method. However, it's often easier just to create a regular control, and to tell the template to use that instead of doing it behind the scenes.

If you plan to use a FileDrop or FileDropCombo, you must populate an additional template in the Extensions window. Highlight the FileDrop(Combo), then press [Insert]. Depending on your control type, you'll be able to choose one of two helper templates from the list:

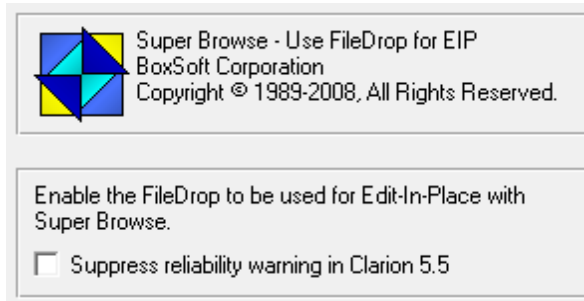


After selecting the template, it will appear below your FileDrop(Combo) in the extensions window:



NOTE: FileDrop and FileDropCombos work with EIP only in ABC, using Clarion 6 and later. The C55 classes do a forced comparison of the original record contents before saving the record, and the support for FileDrop(Combo) confuse this approach, and the wasPrimed property wasn't added until Clarion 6. The result is that the record will be saved only if another field has been changed at the same time as the one associated with the FileDrop. The template will automatically report an error during generation.

We don't believe in changing Clarion's own templates and classes, but you're welcome to do so. If you decide to retrofit this property from Clarion 6 back into Clarion 5.5, the FileDrop(Combo) template will work in C55. Once you've done this, you can suppress the generation error with this switch:



When you're using a FileDropCombo, your USE variable **must** name the corresponding EIP field to be updated. For example, if your EIP field is **Cus:City**, and your FileDropCombo lets them select a record from the **City** file, then the FDC control's USE variable **must** be **Cus:City**, not **Cit:City**.

2.3 Restore Child After Cancel

(Procedure Extension Template)

This template works in conjunction with a BrowseBox of Child records on the same window as the SaveButton for a Parent. The most common example is an Invoice, with the Header being updated, and the Detail records in a Browse.

The problem with this is that any changes made via the Browse immediately change the data file. If the user cancels the update, the children are not automatically restored. This template solves this problem.

This template remembers the children when the form is called. If the user cancels, the current children are deleted and the original children are re-inserted. Any records that have not changed status or value will not be touched.

It's assume that the window contains a SaveButton template for the Parent and a Browse for the Child. The Browse must use a "File Relationship" Range, with the Parent file as the related file.

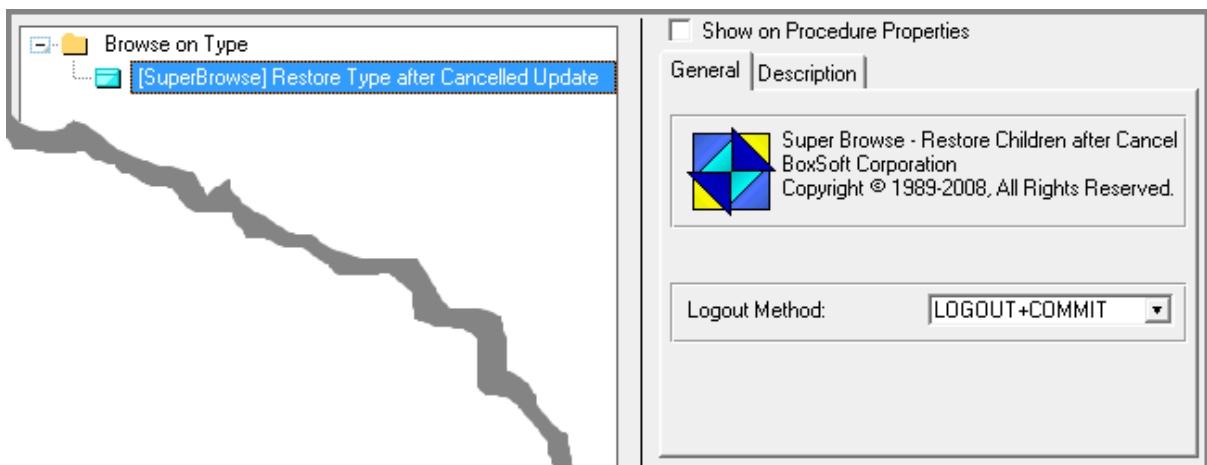
The user must not be allowed to change the Primary Key Field of the Parent file, which connects it to the child!

BLOB fields are not supported in the child file.

Populating the Template

To add this template, simply go to the Extensions window from your Procedure Properties window, *highlight the desired browse box*, then press the [Insert] button to add the template. If you have multiple browse boxes, then you can add this extension to each one. Once you've added the template, you'll see the following settings:

The template automatically matches the BrowseBox with the SaveButton, based upon the default Range settings of your Browse. If it cannot determine the necessary information, it will display an error during code generation.



Logout Method - You may wish to wrap this operation in a transaction. Your options are "LOGOUT+COMMIT" (a regular transaction frame), "LOCK+UNLOCK" (unconventional, but useful sometimes), or "None".

Embed Points

Before Adding Record to Child File - This embed is called before each record is added to the child file. You would use this location to make adjustments to inventory, and other such tasks. For example, you might do the following:

```
Prd:No = Itm:PrdNo
GET(Product, Prd:NoKey)
Prd:InStock -= Itm:Quantity
PUT(Product)
```

Before Deleting Record from Child File - This is the companion embed to the one above. It is called before each record is deleted from the child file. You would use this location to make adjustments to inventory, and other such tasks. For example, you might do the following:

```
Prd:No = Itm:PrdNo
GET(Product, Prd:NoKey)
Prd:InStock += Itm:Quantity
PUT(Product)
```

2.4 Conditional Formats

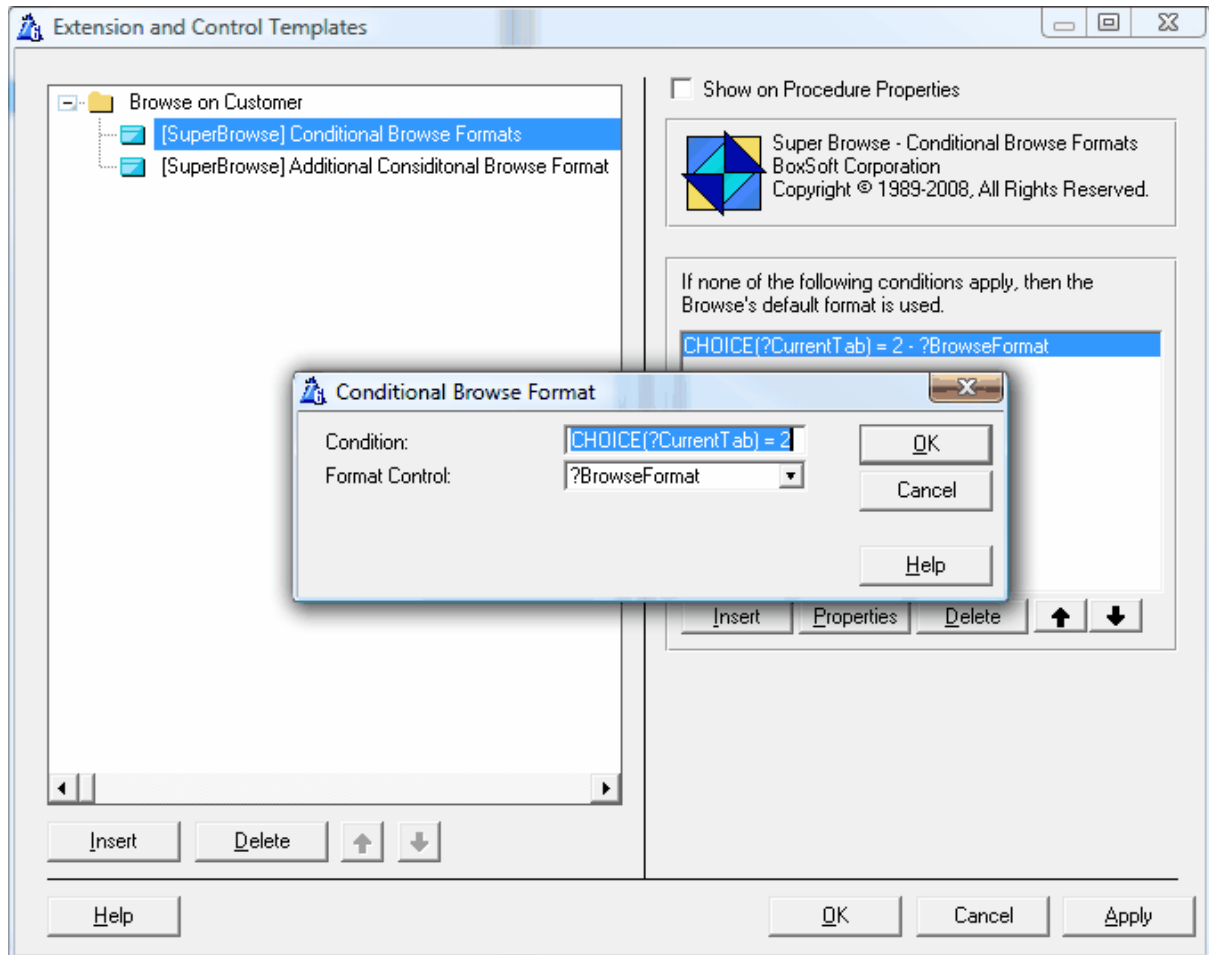
(Procedure Control and Extension Templates)

There are situations where you want different display formats for different conditions. For example, you have a list of customers by last name, so you want to see the last name as the first column. You also view the customers by first name, in which case you want to see the first name column before the others. This template enables you to accomplish this task.

NOTE: Conditional Formats are incompatible with BrowseEntry (Edit In Place) and ActionHeaders. We hope to change this in the future, but it is impossible at this time.

This solution requires the cooperative use of two templates.

1. A control template is used to populate dummy list boxes onto the window. The original list control's format is the "default". Each list box will contain one additional format. When you are adding the dummy list boxes, you can put them anywhere on the window, as they will be hidden automatically when the window opens. They are only place holders to allow to you enter the format information.
2. In the Extensions window, add the "Set Browse Formats" extension template. You can add as many conditions as you like, with each condition requiring the following settings:



Condition - When this expression is true, then the format from the specified control will be used instead

of the default format for the browse.

Format Control - This is the dummy list box hosting the desired format settings.

NOTE: If you want to use colored columns and/or icons, you must be consistent for all formats. If you have the color checkbox ON for a particular column in one format, and that same column exists in another format, then it must also have the color checkbox ON.

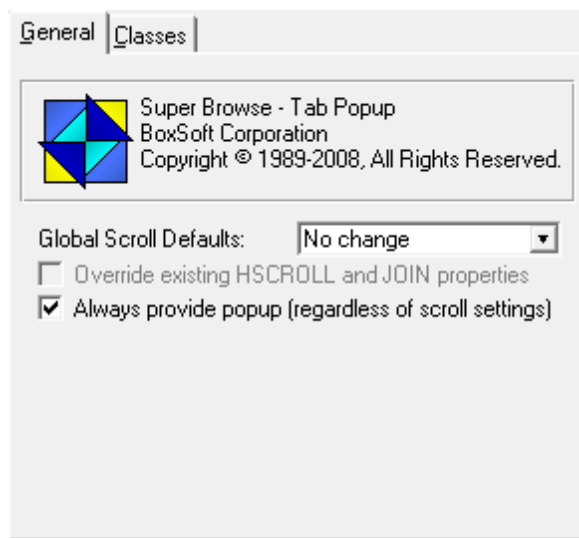
2.5 Tab Pop-ups

(Global Extension Template)

Sheets and tabs are an excellent organizational tool. However, when you get too many tabs on a sheet you have to make sacrifices. You either give up space on your window for multiple rows of tabs, or you use the scroll bars and your tabs get "hidden".

This template creates a tab popup menu for all of the sheets in your application. When your users presses MouseRight over the tab area, a popup menu appears with all of the tab options. The current tab is shown with a checkmark. This gives you the benefit of using many tabs, without losing window space or hiding your tabs beyond scroll buttons.

To implement this template, you add it to your global extensions. The options are as follows:



Global Scroll Defaults - This controls whether scroll buttons are automatically applied to your sheets. There are three settings: No Change, Spread Scroll Buttons, and Joined Scroll Buttons.

Override existing HSCROLL and JOIN properties - If you specify anything by "No Change" for the previous setting, then this controls whether all sheet are overridden, or only those that don't already have scroll buttons. If this is OFF, then only sheet without scroll buttons will get them.

Always provide popup (regardless of scroll settings) - This tells the system to provide a popup menu for the tabs, regardless of whether your tabs have scroll buttons. If this is OFF, then the popup appears only if your sheets have the scroll buttons. (For those of you who like resizing, expanding a window until the all tabs are visible does not remove the scroll buttons.)

2.6 Select Mode Cancel Button

(Global and Procedure Extension Templates)

Clarion allows us to use a browse procedure for both updating a file and selecting records. When we are in update mode, the [Close] button makes sense. When we are selecting records, though, the [Close] button is really a [Cancel] button.

This template automatically changes the text on the [Close] button to "Cancel" when the browse is called to select a record. You start by populating the global extension template. The settings are as follows:

Select Mode Cancel Button - Global Support
This template changes all "Close" buttons to "Cancel" whenever a Browse is called for Selecting.

Original "Close" button text:

Alternate settings during "Select" mode:

Text:

Icon:

Use global icon only when local icon is present.

Original "Close" button text - This is the text that is normally on your close buttons. You may normally change your close buttons to "Exit", so you would put that here. Do *not* include an ampersand (&), as the template automatically strips it when looking through screens.

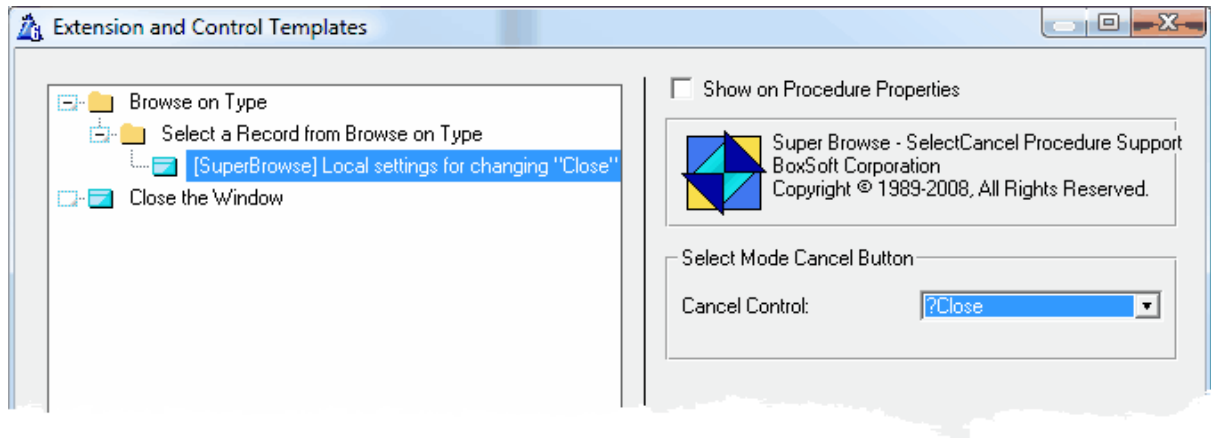
Alternate Text during "Select" mode - This is the text that you want to display on the button when the browse is called to select records.

Alternate Icon during "Select" mode - If you're into icons, this will change the icon to match your standards for the Cancel button.

Use global icon only when local icon is present - If you are not very consistent in your usage of icons, this will apply the above icon only if there is already an icon defined for that button on each window.

Some of you may wonder why we search for the "Close" text on the button, rather than looking for a "Cancel" or "Close" control template. The main reason is that you may use either of these two templates, or you may use a regular button with the "Close Window" or "Cancel Operation" code templates, or your own hand code. Searching for the text should catch most of the buttons.

For those where you've used a different word, you can use the local extension template to tell the global template which button to change on that particular window. This extension template can only be added by first highlighting the Select button extension template in the extensions window.



Cancel Control - This is the button which cancels the select operation on this particular window.

2.7 Action Headers

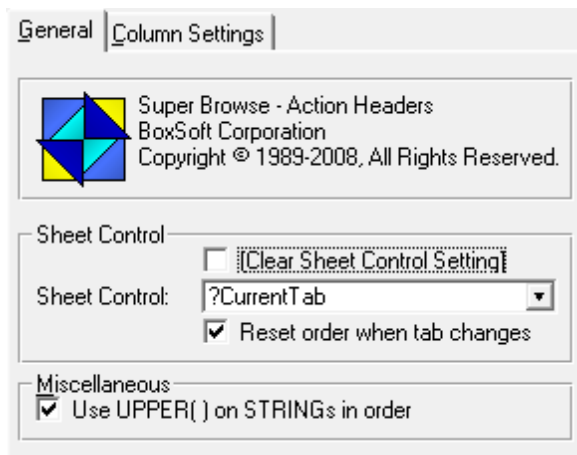
(Procedure Extension Template)

The Windows 95 Explorer utility and many other database programs have popularized the ability to press the header of a list box to sort the column in that order. This extension template allows you to achieve this feat. If you are using a custom sort order and you press the header a second time, the order is reversed.

You can use the headers to select your regular Tabs (to use the built-in keys), or you can use a custom order of one or more fields. You can also execute your own source code.

To add this template, simply go to the Extensions window from your Procedure Properties window, *highlight the desired browse box*, then press the [Insert] button to add the template. If you have multiple browse boxes, then you can add this extension to each one. Once you've added the template, you'll see the following settings:

General Tab



Sheet Control

[Clear Sheet Control Setting] - This just clears the following setting, because once you've chosen a control, you cannot unchoose it directly. Hence this indirect approach.

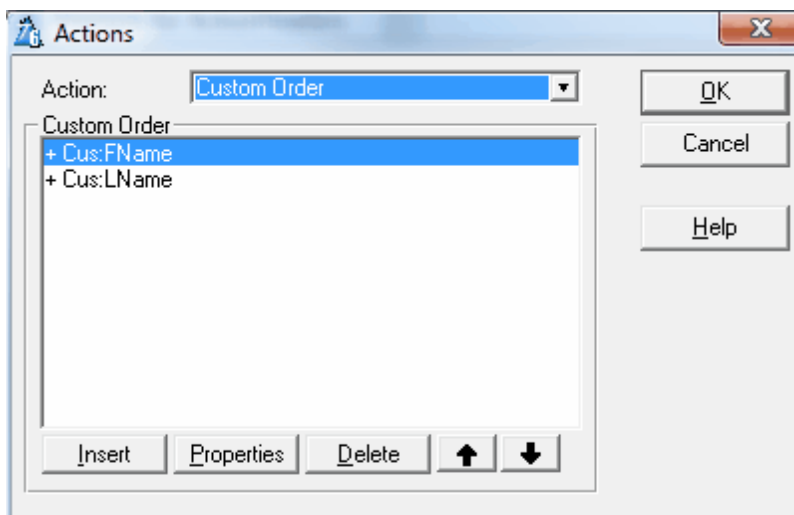
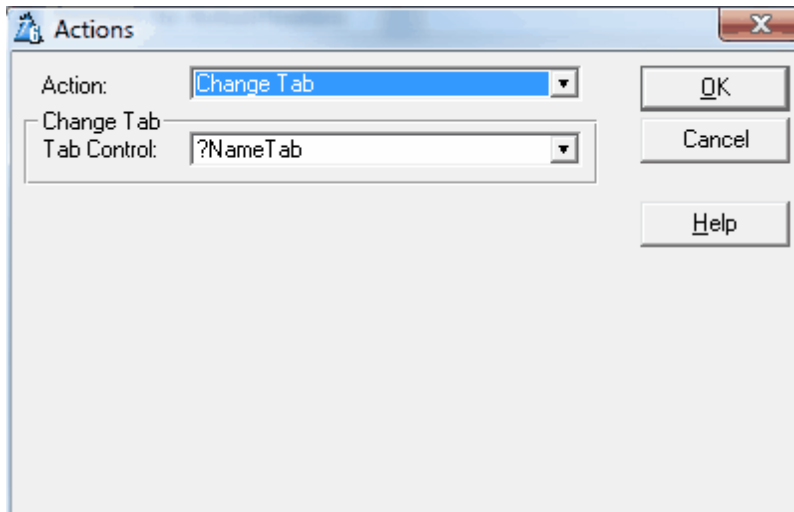
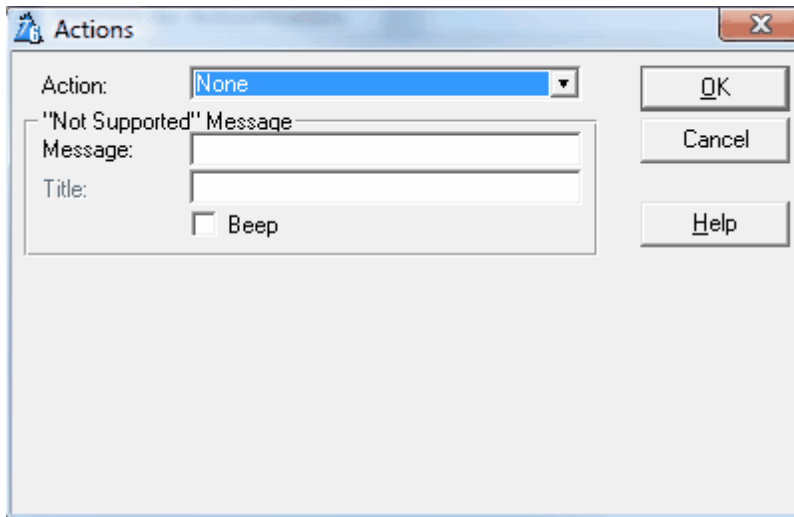
Sheet Control - If you will using the "Change Tab" option, then you just select the sheet control that contains the tabs. If this is a wizard-generated Browse procedure, then it will probably be "? CurrentTab".

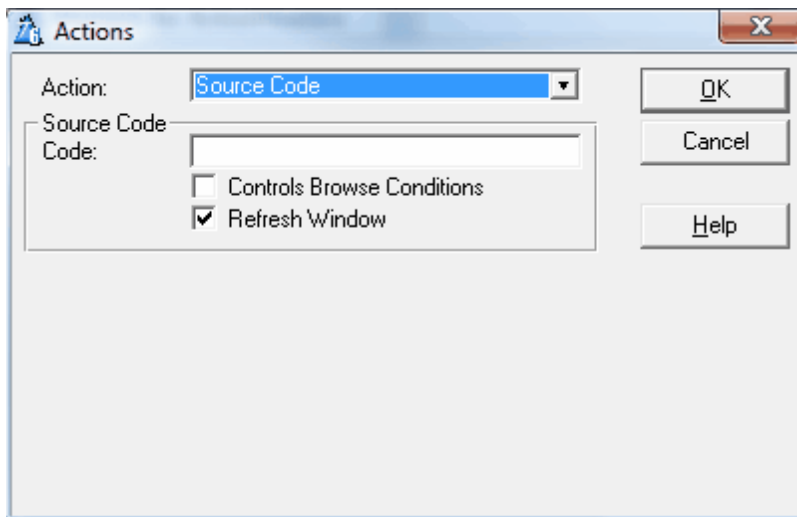
Reset order when tab changes - If you're using the tab to control order, then you may want to force the browse to refresh when the tab changes. Usually the default of TRUE is appropriate.

Miscellaneous

Use UPPER() on STRINGs - When creating the PROP:Order, should the UPPER function be used to ensure case-insensitive sorts. Be careful that your keys and database back-end correspond with this setting.

Column Settings





Each column sorts itself - To quickly apply sorting to columns, based on their basic contents, turn this ON. In many situations, however, you'll want to set the sorting options individually for each column.

For each column in your Browse box, you will see an entry in the following list. You can optionally add the Action Header support to any of these. When you press [Properties] for a particular column, you will see the following settings:

Action - This can be *None*, *Change Tab*, *Custom Order*, or *Source Code*. Depending on what you select, different options will be available:

"Not Supported" Message - If you specify *None* above, then you can enter message text to be displayed when the user presses that header, and/or you can have it BEEP.

Tab Control - If you chose *Tab Control* above, then you must choose the desired tab control here. If you are using a wizard generated browse box, then your tabs will probably not have control equates. Therefore, you must go into the Window formatter and assign equates to the tabs before you can select them here (e.g. ?NameTab, ?CityTab, etc.).

Custom Order Segments - If you chose *Custom Order* above, then you can add any number of order segments here. For each segment, you may choose the field to be ordered, plus the sequence (*Ascending* or *Descending*).

These fields can be any field that is displayed in your list or any hot field (regardless of whether it is from the primary file). Be careful, though, because this feature uses `View{PROP:Order}` (through the `ViewManager`), which will be slow for a large file using a non-keyed field.

At this time, Custom Orders are not compatible with Locators. This should not be a problem, however, because you will normally be using this on a browse with a small subset of records where a locator is not crucial. Otherwise, `View{PROP:Order}` can become quite slow. The locator will not work while you are using a custom order. When you switch back to a keyed order, it will work fine.

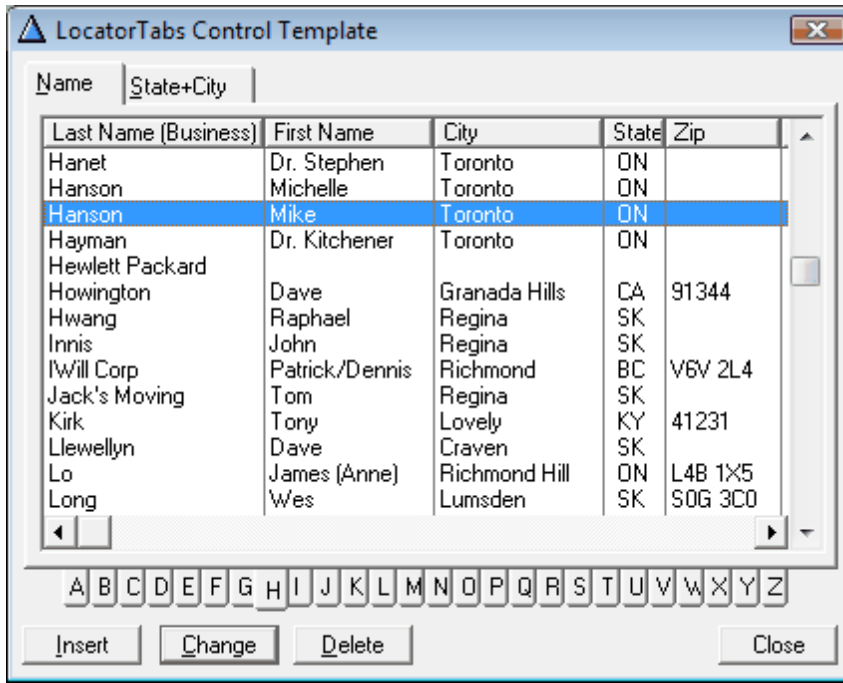
You will also have problems with Custom Orders and scroll bar thumbs. They will not properly reflect the position within the list. Changing them to Fixed Thumbs will help, but they will still act strangely. We are looking into ways to improve both of these shortcomings.

Code - If you chose *Source Code* as your Type of Action, then you can enter any source code here. If

you know that this will require a browse refresh, then turn ON "Controls Browse Conditions". Then it will refresh only the associated browse. Alternatively, you can have the entire window refresh afterward (with `ThisWindow.Reset`). This option should only be used by confidently knowledgeable developers.

2.8 Locator Tabs

(Procedure Control Template)



Many applications have "rolodex-style" locators for their browses (see the Solodex example that comes with CW 2.0.) Some users love to use their mouse whenever possible, and they prefer this visual locator over having to type the locator value.

This control template places a sheet of locator tabs on your window. You can place them above, below, to the left or right of your browse box. They automatically scroll if there is not enough space to show all letters. You can optionally support spaces and numbers.

The tabs use the regular locator code associated with your browse, so if you are not supporting locators, then the tabs will be disabled. If some of your BrowseBox conditions support locators and some don't, then the locator tabs will appear and disappear accordingly.

Regardless of whether your regular locators are step, incremental or entry, the Tab Locators will operate in "Step" mode. Your regular locators will work as they always have.

Using Locator Tabs instead of Locator Buttons is an improvement, as you can easily change the position of a sheet, versus individually positioning many buttons. (e.g. Clarion's Solodex example uses locator buttons.)

This is a control template, so you must populate it onto your window. Go into the window formatter, and press "Populate / Control Template" from the pulldown window (or press the bottom right button on the populate toolbox).

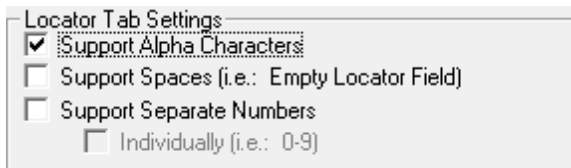
The template requires Clarion's BrowseBox. If you don't already have one on your window, then the "LocatorTabs" template will *not* be available for selection. If you have only one BrowseBox, then it will automatically be selected. If, however, you have more than one BrowseBox on your window that does not yet have LocatorTabs associated with it, then you will be offered a list of available browse boxes with which you can associate this new control template.

NOTE: When you first populate the template, a bug in Clarion causes it to add a tab control along with the sheet. Leave this in place, as it makes it easier to position the sheet. The template automatically hides the extraneous tab at run-time.

The initial orientation of the sheet is vertical, to be placed on the left side of the browse box. If you wish, you can change it to the top, bottom, or right side. This is done with in the Properties window of the SHEET control. If you tell it to use top or bottom, then you will have to adjust the "Tab Width". If your window uses MS Sans Serif (8 pt.), then a good value is "9".

If you know that all tabs will appear without scrolling, then you may want to remove the scroll buttons from the sheet. If you are also using the TabPopup global template, then these tabs will be supported along with everything else.

If you look at the "Actions" window (or go to the Extension window from the Procedure Properties), then you will see the following settings:



Support Alpha Characters - If your key field is numeric, then turn this off to suppress the letter tabs. In this situation, you should also turn ON the "Numbers Individually" option below..

Support Spaces (i.e. Empty Locator Field) - If any of your records can have empty locator fields, then you should turn this on to allow your user to get to the spaces at the top of the list. Otherwise, they will always start with "A". The Space locator appears as a blank tab.

Support Separate Numbers - If any of your records contain locator fields starting with a numeric digit, then you should turn this on to allow your user to get to the numbers at the top of the list. Otherwise, they will always start with "A".

Individually (i.e. 0-9) - If you want each digit represented as a separate tab, then turn this on. Otherwise, a single tab will appear with the "#" symbol, which will locate to "0".

2.9 Color List Selector

(Global and Procedure Extension Templates)

A confusing issue for users involves the list selector bar. Regardless of whether your list box has focus, the highlighter bar is very bold. It is difficult to see the insignificant dotted line indicating which control has focus. The problem is increased if you have multiple list boxes on the same window. This leads to the user frequently tabbing around the window just to figure out where they are.

This extension template changes the selector bar color when the list box does not have focus, making the control with focus much more obvious. You can use this feature on any list box (not just Clarion's BrowseBox). If you happen to be using Clarion's BrowseBox, though, it will also dim the highlight bar when you are in the update form.

The ABC version use a combination of a global template and optional procedure template for overriding the global settings. In contrast, the Clarion/Legacy version has only a procedure template.

Global Template (ABC only)

To add this template, go to the Global Extensions window, and press the [Insert] button to add the template. You'll see the following settings:

Use only when local template is also present - Turn this ON if you want the colored selector only in certain special procedures, but not otherwise. You will have to add the local extension to that procedure to specify your settings.

Use color dialog to select colors - If you prefer using the color dialog instead of a drop list of common colors, then turn this ON.

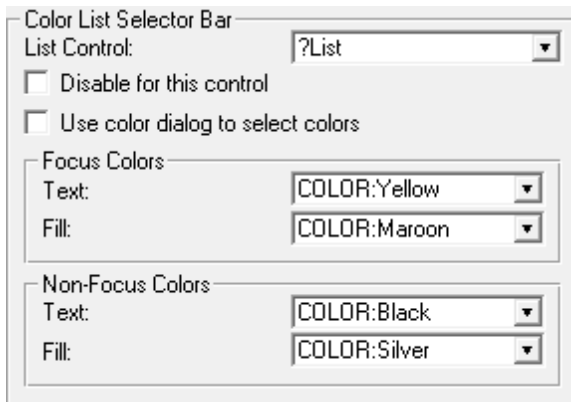
Focus Colors - These are the text and background colors for the list box selector bar when it is the current control. (You could, instead, set these in the Window formatter, but it's probably easier to set them here.) If you leave either of these settings as "COLOR:NONE", then the specified attribute will not be changed.

Non-Focus Colors - These are the text and background colors for the list box selector bar when it is *not* the current control. If you leave either of these settings as "COLOR:NONE", then the specified attribute will not be changed.

Local Template (ABC and Clarion/Legacy)

Once you have added the global extension, you may want to override your settings for one or more

special procedures. In this case, go to the Extensions window for the desired procedure, and press [Insert] to add the template to that procedure. (If you have multiple list boxes to override, add the template multiple times.) Many of the settings are the same as the global extension. The exceptions are:



The image shows a dialog box titled "Color List Selector Bar". It contains the following settings:

- List Control: ?List (dropdown menu)
- Disable for this control
- Use color dialog to select colors
- Focus Colors:
 - Text: COLOR:Yellow (dropdown menu)
 - Fill: COLOR:Maroon (dropdown menu)
- Non-Focus Colors:
 - Text: COLOR:Black (dropdown menu)
 - Fill: COLOR:Silver (dropdown menu)

List Control - This is the LIST control that you wish to affect. This will be something like ?Browse, ?List, ?Browse:1, etc.

Disable for this control - By turning this ON, you will tell the global template to ignore this list control.

2.10 Bold Selected Tab

(Global and Procedure Extension Templates)

When we watch users, we often notice that they are confused as to their current tab. They will often change tabs, then change back again, just to be sure that they are in the right place.

This extension template will highlight the currently selected tab using the bold attribute, text color and/or tab color. (You can use this for non-browse sheets too.)

The ABC version use a combination of a global template and optional procedure template for overriding the global settings. In contrast, the Clarion/Legacy version has only a procedure template.

Global Template (ABC only)

To add this template, go to the Global Extensions window, and press the [Insert] button to add the template. You'll see the following settings:

Use only when local template is also present - Turn this ON if you want the bold tabs only in certain special procedures, but not otherwise. You will have to add the local extension to that procedure to specify your settings.

Use BOLD font attribute - If you turn this on, then the bold attribute will be turned on when the tab is current, and turned off when it is not.

Use color dialog to select colors - If you prefer using the color dialog instead of a drop list of common colors, then turn this ON.

Tab Text Color - If you are changing the tab's background color, then you will probably need to change the color of the text too. Your original text color will be saved before it's changed, and restored after. If you leave this setting as "COLOR:NONE", then the attribute will not be changed.

Tab Fill Color - This is the background color for the tab. The entire sheet rectangle is colored, not just the tab itself. Again, the original tab color will be saved before it's changed, and restored after.

Local Template (ABC and Clarion/Legacy)

Once you have added the global extension, you may want to override your settings for one or more special procedures. In this case, go to the Extensions window for the desired procedure, and press [Insert] to add the template to that procedure. (If you have multiple sheets to override, add the template multiple times.) Many of the settings are the same as the global extension. The exceptions are:

Bold the selected Tab
Sheet Control: ?CurrentTab
 Disable for this sheet

Override Global Bold Setting
 Use BOLD font attribute

Override Global Tab Color Settings
 Use color dialog to select colors

Tab Colors
Text: COLOR:White
Fill: COLOR:Purple

Sheet Control - This is the SHEET control that you wish to affect. If this is a wizard-generated Browse procedure, it will probably be "?CurrentTab".

Disable for this sheet - By turning this ON, you will tell the global template to ignore this sheet control.

Override Global Bold Setting - Turn this ON to override the default global setting. (The global setting is either ON or OFF, and this setting allows us to say "just leave it alone".)

Override Global Tab Color Settings - Turn this ON to override the default global settings. (The global setting is either ON or OFF, and this setting allows us to say "just leave it alone".)

2.11 Display Locator

(Procedure Control Template)

This template displays the current value of the Incremental, Filter, and Step locators. This enables your users to see the locator value as they type it.

This is a control template, so you must populate it onto your window. Go into the window formatter, and press "Populate / Control Template" from the pulldown window (or press the bottom right button on the populate toolbox).

The template requires Clarion's BrowseBox. If you don't already have one on your window, then the "DisplayLocator" template will *not* be available for selection. If you have only one BrowseBox, then it will be selected automatically. If, however, you have more than one BrowseBox on your window that does not yet have LocatorTabs associated with it, then you will be offered a list of available browse boxes with which you can associate this new control template.

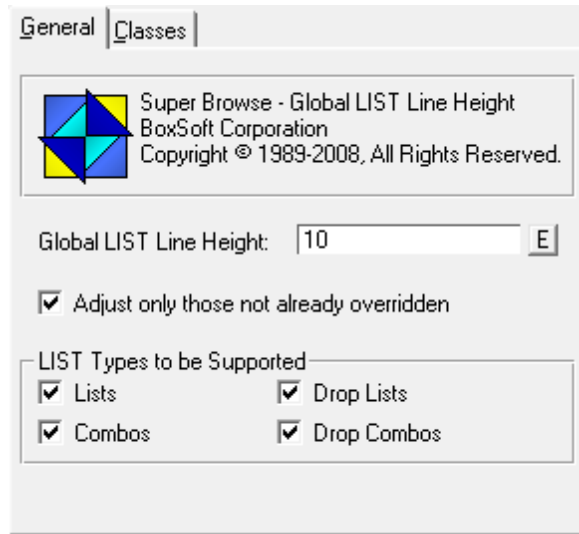
The default width of the control is large enough to hold about 7 characters. You should expand the control's width if you anticipate that your users will type more than this into the locator.

There are no additional settings for this template. It derives all necessary information from the Browse settings.

2.12 Global LIST Line Height

(Global Extension Template)

This global template adjusts the line height in all the windows in the APP. It's similar to the "LIST Line Height" setting in the browse template.



Global LIST Line Height - This is the desired line height for LIST controls throughout your application. Enter a number or any numeric expression here. The default is 10, which we find to be much better the standard of 8.

Adjust only those not already overridden - You may want to have some LIST controls with a different line height than the global setting. In that case, set that individual line height as soon as the window opens (or use the regular Browse settings), and this template will not override your custom settings.

LIST Types to be Supported - This template can change the line height for all LIST control types. You may not want this adjustment made to all types, so you can adjust the selectivity here.

3 Appendices

3.1 Example Programs

There are two example programs provided, one each for ABC and Clarion/Legacy. Check the Installation topic for their location.

There are a number of procedures in each application, individual ones for some of the major features, and a single procedure containing "most" of the features (as many as we could reasonably fit in one place).

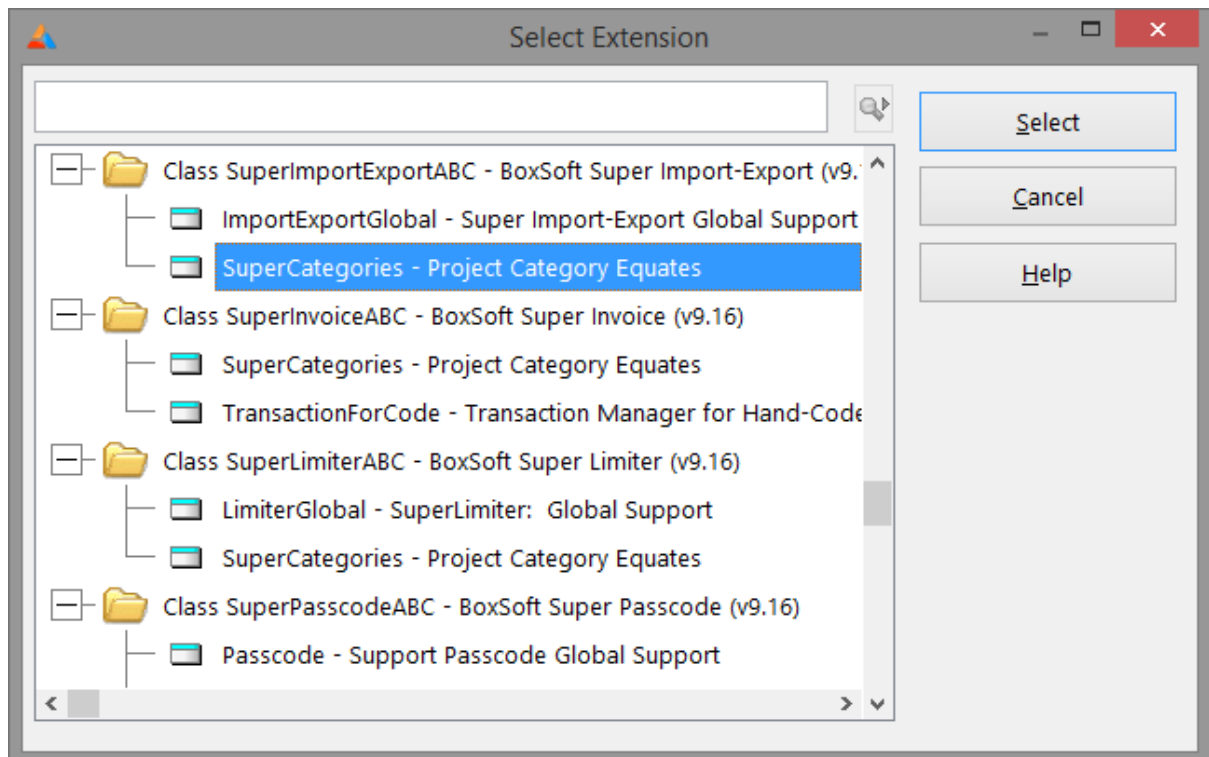
One key difference between the "Single Feature" procedures to the "All Features" procedure is in the Action Headers. If you look at the ActionHeaders procedure, you will see an example of Custom Orders. Because this are not compatible with Locators, we are using only the Change Tab option in the "All Features" procedure.

3.2 Project Defines

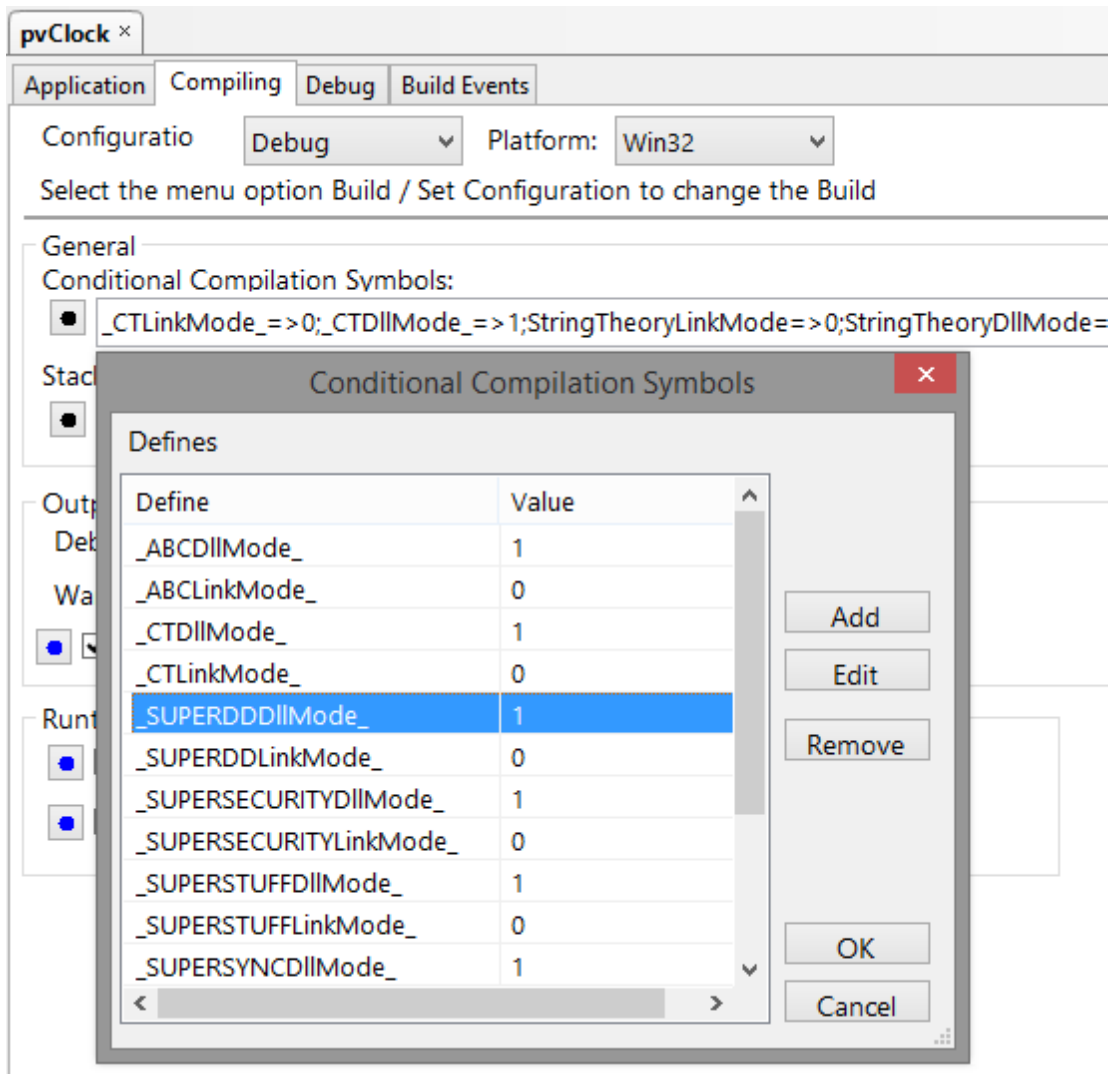
Prior to our 7.0 template versions, we were utilizing the same LINK and DLL Project Defines as the ABC libraries: `_ABCLinkMode_` and `_ABCDIIMode_`. This caused all of our libraries to be included in your base/dictionary DLL, even if you weren't using them in a particular development project.

To make this inclusion more selective, as of our 7.0 versions we changed to use various other switches. Some of these are product related (e.g. Super QBE uses `_SuperQBELinkMode_` and `_SuperQBEDIIMode_`), while others are associated with one of our shared base classes (e.g. Drag & Drop uses `_SuperDDLLinkMode_` and `_SuperDDDIIMode_`). Usually the templates (especially the global ones) automatically add the necessary entries to your Project Defines. If you happen to use the templates in your APPs in the wrong combination, these can be inadvertently omitted.

For APP-based systems, you can force the switches to be included by using the Super Categories global extension template. Every one of the Super Templates has this extension to apply its own switches, so if you're using multiple templates in a particular APP, you may have to add this extension for each of the products. (As was mentioned above, if there's already a global extension populated for a given Super Template, then you don't have to add this extension for that product.) Even if it's not needed, there's no problem with adding the SuperCategories global extension.



For hand-coded PRJ-based systems, you must add the switches manually. Take note of their names in the INC files, and then add them to the project settings like this:



3.3 Troubleshooting

Problem:

I am trying to populate an extension template, but I can't see it in the list under SuperBrowse.

Solution:

First, make sure that you are not trying to populate a control template in the Extension windows, or an extension template in the Window formatter. (Yes, it is confusing that you can access Control template Actions from the Extensions window.)

Some of the extension templates require Clarion's BrowseBox template. When you try to populate one of these extensions, you must first highlight the desired BrowseBox. If you don't have a BrowseBox, then you must first add one to your window. If you already have one, check to see if you've already added the desired extension to the browse box.

Problem:

My field validation code is not being executed when the user is scrolling through a different column.

Solution:

Validation code is only applied for columns which are entered and exited. To validate a field, regardless of whether it is changed, go to [Global], [Embeds], [All Embeds], [Contract All], [Field-level Validation], etc. This code will be executed whenever a record is added or updated. The proper syntax is:

```
IF NOT NecessaryCondition
  RETURN Level:Notify
END!IF
```

Problem:

When I compile, I'm getting errors in my Edit-In-Place procedure regarding duplicate labels, already defined, and things like that.

Solution:

Go into your column settings and verify that each of your "Object Names" are unique.

Problem:

The thumbs in the vertical scroll bar are acting strangely.

Solution:

If you are using Custom Orders with Action Headers, then the thumb will often not work properly. Changing to a fixed thumb will help, but will not entirely solve the problem. We are looking into ways to solved this problem.

Problem:

During code generation of a procedure with ActionHeaders, an error occurs in which %ListControl is undefined.

Solution:

This is a bug in CW. It's storing the ActionHeaders template in the APP before the BrowseBox. There are a variety of tokens that ActionHeaders needs that are defined in BrowseBox, which causes this error. You can fix this with the following method:

1. Backup your APP.
2. Perform a "Selective Export" of the problem procedure.
3. Edit the TXA file. Cut into the clipboard from the section starting with:

```
[ADDITION]  
NAME SuperBrowse ActionHeaders
```

and ending with (but not including) the next [ADDITION] line.

4. Paste this section after the BrowseBox [ADDITION]. (This means immediately before the additional following the BrowseBox.)
5. Load your APP and Import the TXA. It will ask you to "Rename, etc.". Tell it to "Prompt". Then tell it to "Replace" the browse procedure. For each procedure the browse calls, you will see a "replace/rename" option. This is very important: Press [Esc] for each of these. DO NOT choose the buttons. By pressing [Esc], you are telling it to call the same procedure.

Problem:

I'm getting all kinds of errors with my dimensioned fields.

Solution:

There are numerous problems with dimensioned fields and CW templates. We don't believe in using them anyway, so we strongly suggest that you avoid them in your data files. If you really need them, and you wish to use them in your browses, then create a GROUP+FIELDS your file that is OVER the dimensioned array. Then you can refer to these variables instead.

Problem:

I'm trying to use a FileDropCombo with BrowseEntry (Edit-In-Place), but the value that appears in the FDC doesn't match the current value of my EIP field, and values chosen with the FDC are not saved to the EIP field.

Solution:

When you're using a FileDropCombo, your USE variable must name the corresponding EIP field to be updated. For example, if your EIP field is Cus:City, and your FileDropCombo lets them select a record from the City file, then the FDC control's USE variable must be Cus:City, not Cit:City.

3.4 Contacting Technical Support

If you have any troubles with this product, then please contact:

Mitten Software
2354 West Wayzata Blvd
Second Floor, Suite H
Long Lake, MN 55356

Voice: (952) 745-4941
Fax: (952) 745-4944

Internet: www.mittensoftware.com
answers@mittensoftware.com
www.boxsoft.net
www.boxsoft.net/contact.htm

3.5 License Agreement

One License per Developer

This Super Template product is comprised of the templates, default applications, libraries, source code, documentation, and help files provided with the package. You must have a separate registered copy for each developer using it.

Redistribution

You are allowed to use the product for any programs that you create, and you are permitted to distribute the generated source code. You may not, however, distribute any portion of the product in its original or modified form without the prior written consent from BoxSoft Development.

One exception to this is the example programs provided with this installation or separately from BoxSoft or its agents: these may be distributed without penalty, in either their original or a modified state.

Disclaimer

BoxSoft Development does not warranty this software for any use. Any expenses or lost time due to errors in this product are not the responsibility of BoxSoft Development. We will attempt to fix any errors that are brought to our attention, but we are not legally liable for any lack of correctness of the product.

Index

- A -

ABBROWSE.CLW 3
Action Headers 31
Adding SuperBrowse to your Applications 12

- B -

Bold Selected Tab 39
Browse Entry 13

- C -

Cancel 24
Cancel Button 29
Child 24
Clarion Source Code Modifications 3
Class Libraries 7, 44
Close Button 29
Color 37, 39
Color List Selector 37
Column Headers 31
Conditional Format 26
Contacting Technical Support 48
Control Template 13, 35, 41
Conversion 11
Custom Orders 31

- D -

Defines 44
Descending 31
Directories 7
Disappear 44
Disk Set 11
Display Locator 41
DLLMode 44

- E -

Edit-In-Place 13
EIP 13

Entry 13
Example Program 43
Extension Template 24, 26, 28, 29, 31, 37, 39, 42

- F -

Format 26
Freeze 44

- G -

Global 42
GPF 44

- H -

Headers 31
Highlight Color 37

- I -

Include Files 7
In-Line Entry 13
Installation 7
Introduction 3
Invoice 24

- L -

License Agreement 49
Line Height 42
LinkMode 44
Locator 41
Locator Tabs 35

- M -

Memory Set 11

- O -

Order 31

- P -

Pop-up 28
Project Defines 44

- R -

Redirection File 7
Registering the Template 7
Restore Child After Cancel 24
Reverse Order 31
RTFM Warning 6

- S -

Scrolling Entry 13
Select Mode Cancel Button 29
Selector Color 37
Sheet 28
Sort 31
Spreadsheet Entry 13
Support 48

- T -

Tab 39
Tab Color 39
Tab Pop-up 28
Tabs 35
TagFile_ (POINTER) 11
TagFilePos_ (POSITION) 11
Tech Support 48
Template Registry 7
Transaction 24
Troubleshooting 46

- U -

Upgrading 11

- W -

Worksheet Entry 13