

**BoxSoft**  
Corporation

## **Super Stuff (MH) - Documentation**

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

# Table of Contents

Foreword	0
<b>Part I Getting Started</b>	<b>4</b>
1 Introduction.....	4
2 RTFM Warning!!!.....	6
3 Installation.....	7
<b>Part II Template Usage</b>	<b>11</b>
1 Adding SuperStuff to your Applications.....	11
2 Browsers.....	12
Threaded Browse+Form .....	12
Wrap Browse Field .....	15
Toggle "Active" Field in Browse .....	16
Conditional Browse Formats .....	20
Select/Cancel Support .....	21
EntryLocators respond to EnterKey .....	22
Hot Fields for Browse .....	22
3 Reports/Processes.....	24
Ask Date Range for Report .....	24
Use a Non-Default Printer for the Report .....	25
Export Records to CSV File using Process .....	25
Memory Report .....	26
Hot Fields for Report or Process .....	27
Pre-scan RecordsToProcess .....	27
Hide ProgressWindow .....	28
4 TABs.....	29
Initially select TAB other than first .....	29
TabPopups for all SHEETs .....	29
Hide TABs During Insert .....	30
5 Enhanced Entry.....	31
Drag & Drop (MANY:MANY) .....	31
Enter Is Tab .....	32
Quicken Date Entry .....	34
Limit Procedure to One Thread .....	34
Trigger a Calendar Lookup .....	35
Drop Buttons .....	36
Drop Menus .....	42
6 Resize Support.....	45
Introduction to Resizing .....	45
Resizing Global Support .....	46
Measure Frame for Resizing .....	50
Resize Window .....	50
Update Control's Position for Resize .....	59
Split Window (Horizontal and Vertical) .....	60
7 Miscellaneous.....	62

Build Custom VIEW .....	62
Add Open+CloseFiles to Source Procedure .....	69
Restore Child After Cancel .....	70
Look for Children records .....	71
Select default icon for the entire app .....	74
Animated Icon .....	74
Prototype Exporter .....	76
Make Empty *.TPE Files .....	76
Call Procedure with "Cancel" Support .....	77
Embed Point Guidance .....	79
Add Procedure to ThisWindow.Run .....	79

**Part III Appendices 81**

1 Examples.....	81
2 Project Defines.....	84
3 Troubleshooting.....	86
4 Contacting Technical Support.....	88
5 License Agreement.....	89

**Index 90**

# 1 Getting Started

## 1.1 Introduction

Let's start with a bit of history. These templates have evolved steadily over the past years (ever since the release of Clarion for Windows 1.0). They include an interesting mix of features, some of which were developed in association with magazine articles that I authored, and others that I created for my own personal uses. Consequently, it's an odd mix of tools. Some of them, like mhResize and mhView are quite involved, while others (like mhCalendarLookup) are quite simple. Up until now, they've been free.

In general, they were the loose bits that didn't fit anywhere else. Occasionally I would include one here and in one of the commercial Super Templates, where appropriate. Because none of them seemed significant enough to be split off into an independent commercial entity, they languished in the "MikeHanson" template set, with no documentation and little support.

This was a bit of a quandary. I wanted to document and support these templates better, but I could not do it at the expense of one of my paying products. Additionally, I realized that there were so many features included in this set, that it had become a worthwhile product unto itself.

So now I had to decide on a name. Since the template chain was called "MikeHansonABC", I briefly toyed with calling them the "SuperMikeHanson" templates. That seemed a little too narcissistic. Then I considered "Super OddsAndEnds", but that seemed too awkward. Finally, I settled on "Super Stuff". It was quick, cute, and had a nice alliteration.

For technical reasons, the template chain will still be MikeHansonABC, with template names usually prefixed with "mh". I'll continue using this convention, just so that you can distinguish them from the rest of the templates in your development arsenal. Hopefully this dual-naming convention does not cause too much confusion.

As is the case with all of the Super Templates, the features have been developed with the following goals:

1. It 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.

You'll find an amazing collection of tools in this set. Because many of them were developed for my personal use, you'll get a better impression of how I like to work than you might with some of my other Super Templates (which are often outfitted with other people's needs in mind). Don't worry, though; I never make something so specific for my purposes that it's useful only for me. I have always developed programming tools with an eye towards generic use. That's why I started creating templates (originally models) in the first place.

I hope you enjoy using the templates as much as I did creating them. Feel free to wander through the documentation and discovered all of the gems in this collection. I think you'll be suitably impressed. It should also make you much more productive.

### **ABC and Legacy Template Chains**

This documentation pertains to both the ABC and Clarion Legacy template chains. In some situations we have implemented features for one chain that are not in the other, primarily due to our own needs and customer demand. In other situations, some features were not really applicable to both chains, or would be very difficult to implement.

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, or vice versa, and we will see if your needs can be accommodated.

## 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](http://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.

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

## 2 Template Usage

### 2.1 Adding SuperStuff to your Applications

Some of the following features are achieved through the use of a single template in one procedure, while others require a global template for support. Yet others require templates populated into a bunch of places to get the desired effect. There are also a couple that require that you import one or more TXAs into your application. If you follow the instructions for each of the features, you won't go wrong.

## 2.2 Browsers

### 2.2.1 Threaded Browse+Form

#### *(Global and Procedure Extension Templates)*

Normally Clarion applications use a strict Browse+Form metaphor that verges on SDI: When the user updates a record from the Browse, then cannot return to the browse without completing the update operation and exiting the Form. There are times when your users might want to select one record to update, and then (without exiting the form) return to the browse to edit another. Both update forms would be visible and updated concurrently, on separate threads. The user could return to the browse and update more records, to their heart's content.

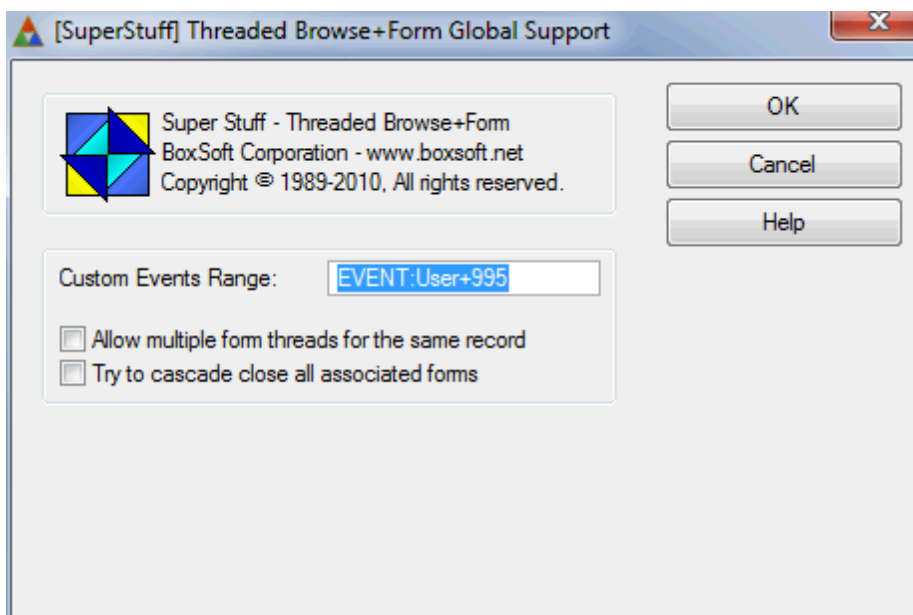
When the update operation completes, the Form signals the Browse, and it refreshes its list accordingly. If the user were updating with any form, it would not work any differently than a regular "SDI" Browse+Form, and the browse would automatically move (as necessary) to highlight the recently updated record. In fact, as long as the currently highlighted record in the browse is the same when the form is closes as it was the form was called, then the highlight will "follow" the update. Otherwise, the browse page is merely refreshed to reflect any possible changes on the currently visible page.

Additionally, if the same browse procedure is running on multiple thread instances, then an update to any of those browses will cause all the browse instances to refresh their pages!

To implement this feature, you must use three templates:

#### **Global Extension**

Just like it sounds, you must add this extension to your global settings. The name of the template is mhThreadedBFGlobal.



**Custom Events Range** - This is the starting value for a small series of internal custom user events. The default is EVENT:User+995.

**Allow multiple form threads for the same record** - Should the user be permitted to update the same record on multiple simultaneous threads? If not, then all additional attempts will be directed to the existing form for that record. This global setting can be overridden on each browse.

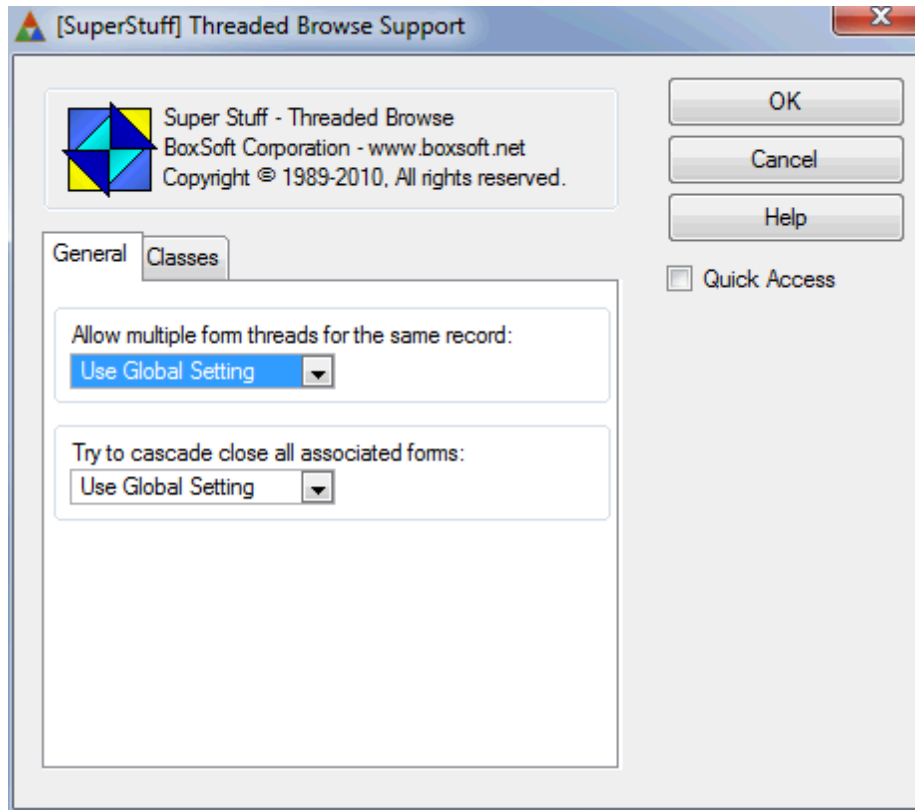
**Try to cascade close all associated forms** - When a browse closes, do you want to post an EVENT: CloseWindow to all open forms for that browse? Note that depending on the state of the form, this may have inconsistent results. For example, if the user is in a window called by the form, then that called window will close, rather than the form itself. Or if the user has made changes to the data in the form, then they'll receive the standard warning for aborting an update.

### **Browse Procedure Extension**

You must add this extension to your Browse procedure. It can be tricky, though. First you must go to the Extensions window, and highlight the existing "Update a Record from Browse Box" template for the desired browse. Then hit the Insert button and find mhThreadedBrowse. If you have multiple browses on the same window, you can add one of these to each browse. The only settings available are the regular class settings, which you can tweak if you feel so inclined.



Here are the template settings:



**Allow multiple form threads for the same record** - Should the user be permitted to update the same record on multiple simultaneous threads? If not, then all additional attempts will be directed to the existing form for that record. This is the local browse override. The default is to follow the global setting, or you can explicitly enable or disable it here.

**Try to cascade close all associated forms** - When a browse closes, do you want to post an EVENT: CloseWindow to all open forms for that browse? Note that depending on the state of the form, this may have inconsistent results. For example, if the user is in a window called by the form, then that called window will close, rather than the form itself. Or if the user has made changes to the data in the form, then they'll receive the standard warning for aborting an update. The default is to follow the global setting, or you can explicitly enable or disable it here.

### **Form Procedure Extension**

You must add this extension to your Form procedure. Just like the Browse template, it can be tricky. First you must go to the Extensions window, and highlight the existing "Update YourFile record on disk" template. Then hit the Insert button and find mhThreadedForm. As with the browse extension, the only settings available are the regular class settings.



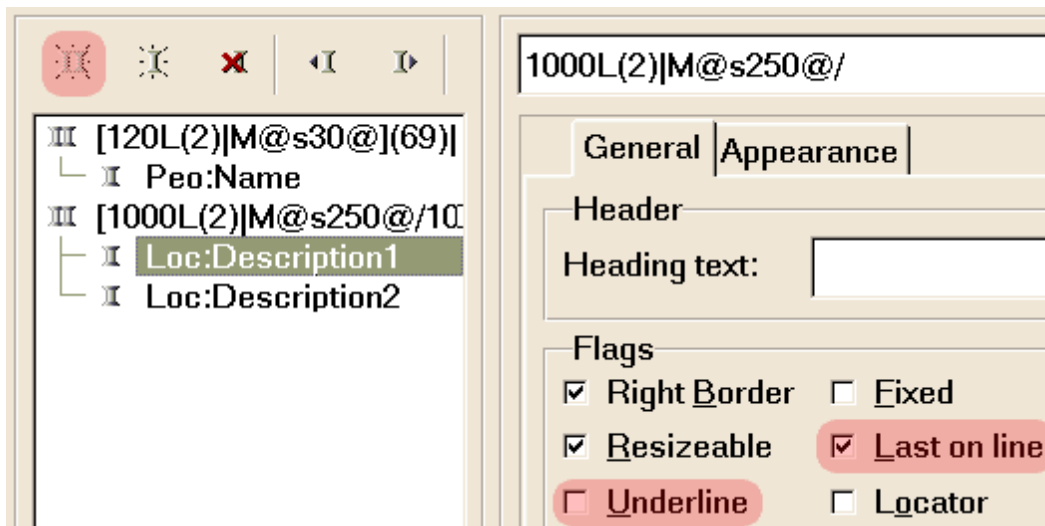
## 2.2.2 Wrap Browse Field

### *(Procedure Extension Template)*

This template splits the contents of a long text field (a big string or a memo) into multiple local variables, so that they can be displayed on multiple lines of your browse box. This wrapping feature makes it easier to read more information on the browse, preventing the hassle of entering the form to see the extra lines.

There are a few steps required to implement this template.

- Create local variables to hold each line segment. We suggest that you make each a STRING(250). They must be at least big enough to hold all visible characters for each row.
- You must place these individual variables in your Browse control. Turn on "Last on line" for all wrap strings except the last. For the Group itself, you will probably want to turn on "Underline", to help the user visually differentiate between each entry in the browse.



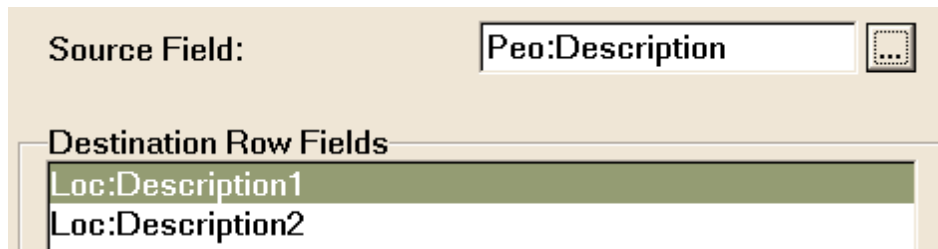
**NOTE:** This template requires the BrowseBox control template to be present on the window. However, this is an extension template that must be added on the Extensions window. Therefore, you must highlight the "Browse on YourTable" template in the Extensions list before hitting Insert to add the template. Otherwise, this template will be missing from the available list. By the way, after you add the template, it will be indented below the Update template in the extensions list, showing its dependency:



You can also have multiple Wrap Browse templates for each browse (because you might have more than one field that you wish to wrap). Each will be shown below its browse, as above.

The template settings are quite straightforward. You must specify the source field (the big one), which needs to be wrapped. Then specify each destination segment field. The template automatically measures the width of the column at run-time along with the font metrics, and fills

each line segment variable with enough characters to fit the line.



One caveat: the template does not automatically re-wrap the text if the user changes the width of the wrapping column. It does, however, auto-adjust, the next time it fills a page of records.

### 2.2.3 Toggle "Active" Field in Browse

#### *(Control Extension Template)*

There are situations where you want to "delete" records without actually deleting them. A common method to do this is to add a field called "Active" to your data file. The default would be "1". In your browse, you implement a filter so that you see only active records. Of course, you also need the option to see inactive entries, and to toggle the state of a record from active to inactive. This template does all this for you.

It adds an option control and button to your window. The options are to see "Active", "Inactive" or "Both". If you are sitting on an active record, then pressing the button will deactivate the record. If, instead, the current record is inactive, then pressing the button will reactivate the record.

Also, the browse's regular "Delete" button will be disabled for Active records. This makes deleting a record a two step process: deactivate it, then delete it. Here are a couple of screen shots:



**Browse People**

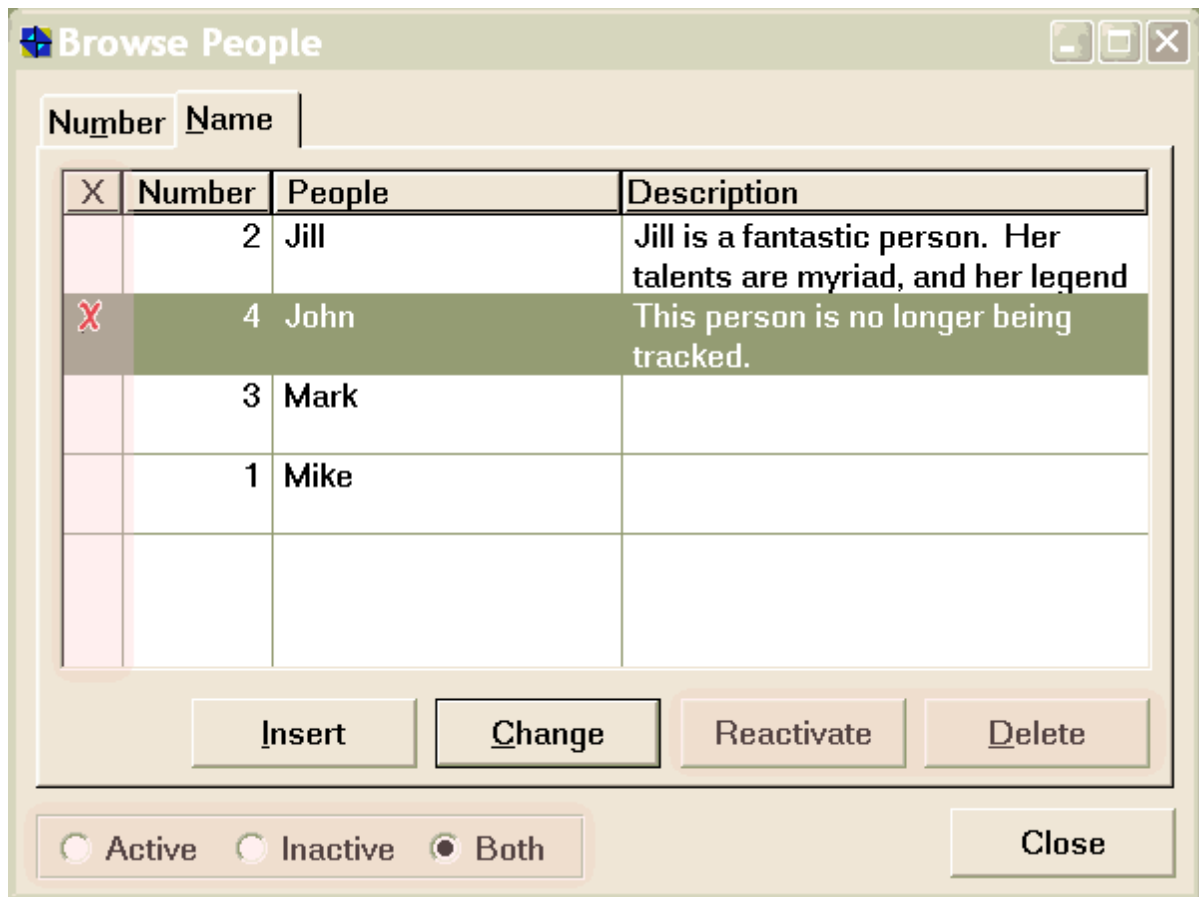
Number Name

X	Number	People	Description
	2	Jill	Jill is a fantastic person. Her talents are myriad, and her legend
	3	Mark	
	1	Mike	

Insert Change Deactivate Delete

Active  Inactive  Both

Close



The OPTION fields uses a local variable called ST::ShowActive, and its possible values are A=Active, I=Inactive or B=Both. The Deactivate/Reactivate button's field equate defaults to ? ToggleActive.

Also note the "X" column. This is your file's Active field, with the Icon attribute turned on (either Normal or Transparent, as you desire). If you would rather display the active status in some other fashion, then just omit the column, and the template will handle it appropriately.

Here are the template settings:

## General Tab

General Display

Super Stuff - Toggle "Active" Field  
BoxSoft Corporation - www.boxsoft.net  
Copyright © 1989-2010, All rights reserved.

"Active" Field to Toggle  
Field:

Custom Code  
Code Before:   
Code After:

Miscellaneous  
Initial State:   
 Do not set filter

**"Active" Field to Toggle** - This is the field in your file to represent the active status (1=Active, 0=Inactive). The template will automatically check your browse's primary file for a field called "Active". If it finds it, it will provide the default.

**Custom Code (Before & After)** - If you want to execute some code when the record's state is being toggled, you can place that code here, or use the embeds.

**Initial State** - Usually you will want to see the Active records when the browse first opens, but this enables you to override that assumption.

**Do not set filter** - The template automatically adds the necessary filters to the browse, so that you see only the appropriate records for the current setting of ST::ShowActive. If you would rather handle the filter creation for this yourself, then turn this ON.

## Display Tab

**Button Text** - These will appear in the ?ToggleActive button, depending on whether the current record is active. Note the use of "&" in the default for "Deactivate". Recall that the browse's &Delete button will be disabled if the current record is active, so the user can use Alt+D to Deactivate or Delete the record, depending on its current state. If your button shows only an icon, then you can specify that the text should appear in the tooltip.

**Question** - This question is asked when the user presses the ?ToggleActive button.

**"Active" Column in Browse** - The template automatically tweaks the "Active" column in your browse. You specify the heading and column width, in addition to the icons. Specify a width of zero to have the width set in relation to your list box's line height. If your heading is actually in the group header (if the "Active" column is alone in its own group), then turn ON the "Use Group Heading" checkbox. The default icons provided with Super Stuff show a blank icon in ACTIVE.ICO, and an "X" in INACTIVE.ICO. Feel free to specify your own icons here.

## 2.2.4 Conditional Browse 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. A control template is used to populate dummy list boxes onto the window. Each list box will contain one additional format. The original list control's format is the "default". Then you add an extension template to specify which format is applied for which condition.

When you are adding the dummy list boxes, you can put them anywhere on the window, as they will be automatically hidden when the window opens. They are only place holders to allow to you enter the formatting information. Once you have specified one ore more extra formats, you can go to the Extensions window.

Then you need to insert 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.

## 2.2.5 Select/Cancel Support

### *(Global and Procedure Extension Templates)*

This global template changes the text and/or icon for a browse's Close button, when that browse is called to perform a Select operation. It automatically looks at all browses in your application, and applies itself accordingly.

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 button text that the template searches for in your browse windows. When checking buttons, it ignores any "&" that it finds.

**Alternate settings during "Select" mode** - This is the word that is substituted during select mode. You can also specify an optional icon. Finally, you can indicate whether the icon should be applied in all instances, or only when a particular Close button already has an icon.

### **Procedure Template (Local Override)**

Cancel Control:

This procedure template allows you to specify which control is used to cancel the selection, in case a particular browse does not have a button with the standard "Close" text, or if its Close button does not cancel the selection.

## **2.2.6 EntryLocators respond to EnterKey**

### ***(Global Extension Template)***

Clarion's Entry Locators expect the user to hit the TabKey after entering the locator string. Many users feel more comfortable with hitting Enter. This global template automatically add EnterKey support to all Entry Locators in an application.

There are no prompts for this template.

## **2.2.7 Hot Fields for Browse**

### ***(Procedure Extension Template)***

When you create Browsers, Processes and Reports, Clarion automatically adds any hot fields that are referenced explicitly in list boxes, reports, template settings, etc. However, in situations where you are using a field in hand-coded source, and you must add that as an explicit hot field. If there are only a couple of fields, then this is not a big deal.

However, you may have situations where there are many fields that are used in your hand-coded source, and adding each of them as individual hot fields is a real hassle. In this situations, these two templates (one for Browsers, and another for Processes and Reports) will automatically add all the file's fields as hot fields.

In a Process or Report, you can just add the extension template as usual. For Browsers, however, it is a bit trickier. First you must highlight the desired browse in the Extensions window, then hit the Insert button and choose mhHotFieldsBrowse.

**One or All** - Is there just one file whose fields are to be projected, or do you want to do it for all the files listed in the tree?

**File** - If you choose "One File" above, then you must specify that file here.

**NOTE #1:** For the sake of compatibility with alternate report templates (e.g. CPCS), you are allowed to add the "Process/Report" version of this extension to any procedure. If you add it where it is incompatible, then there will be an error message during generation.

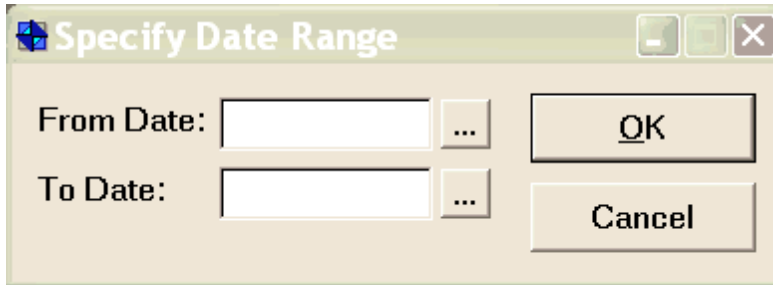
**NOTE #2:** Clarion's Legacy templates do not project dimensioned fields in Browsers, Processes and Reports. Our templates will still add the fields to the QUEUE (in the case of a Browse) and it will compile without errors, but the dimensioned fields will not be projected in the VIEW! If you happen to be using a file driver that automatically passes all fields regardless of whether they are projected, then your program will luckily work. Otherwise, the values of the dimensioned fields will not be fetched from the database. Of course, if you are using dimensioned fields, then you were already destined for hardship. <tease>

## 2.3 Reports/Processes

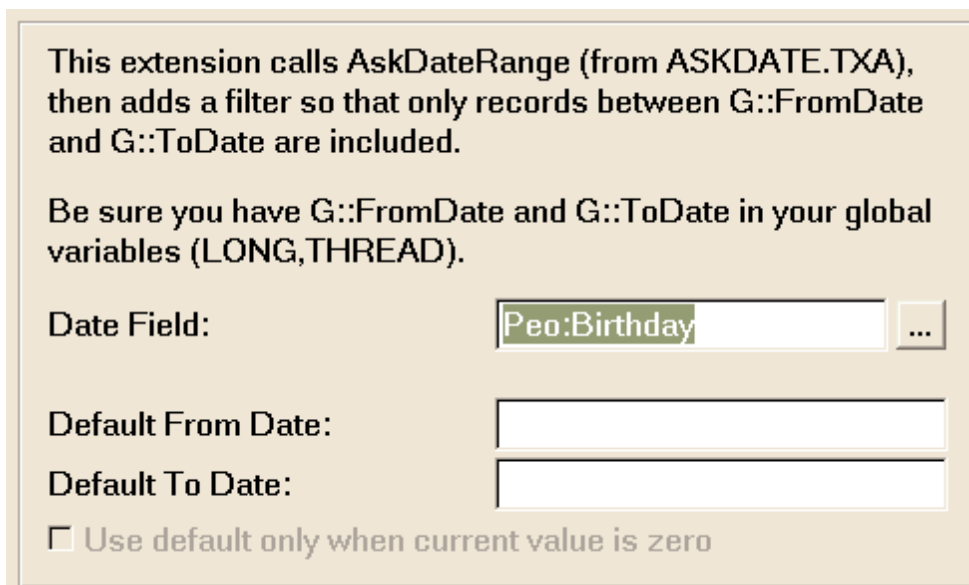
### 2.3.1 Ask Date Range for Report

*(Procedure Extension Template)*

This template adds a date range to a report or process. The user is prompted at run-time, and they can type the start and end dates directly, or use the calendar lookup.



1. Add two global LONG,THREAD variables called G::FromDate and G::ToDate.
2. Import CALENDAR.TXA from CLARION\SUPER\SRC\_???\MHSTF\. This will add the procedure MH::Calendar to your application. If your system has multiple APPs, import this into the dictionary APP, and ensure that it is exported for use by other APPs.
3. Import ASKDATE.TXA from CLARION\SUPER\SRC\_???\MHSTF\. This will add the procedure AskDateRange to your application. For multi-APP systems, follow the same approach as in the previous step.
4. Add the mhAskDateRangeReport extension template to your report.



**Date Field** - This is the field that must be in the date range.



**Default From/To Date** - If you want to provide defaults for the From and To dates (e.g. last year), then put this here. These can be any numeric expression. For example, DATE(1,1,YEAR(TODAY)-1) will give you the first day of the previous year, while the last day of the previous year is DATE(1,1,YEAR(TODAY))-1.

**Use default only when current value is zero** - If you want the defaults to be applied only if G::FromDate and/or G::ToDate are zero, then turn this ON.

### 2.3.2 Use a Non-Default Printer for the Report

#### *(Procedure Extension Template)*

As you know, reports are automatically sent to the default printer. In some cases, you may wish to let you users choose the destination printer each time the report runs. Or you may have a pre-chosen printer stored in a configuration field. This extension template handles either of these situations.

**Prompt for Alternate Printer** - Do you want the PRINTERDIALOG to appear when the report starts? If the user presses Cancel, then the report will be aborted.

**Printer Name Variable** - If you have a variable or field that contains the name of the printer, then specify it here.

**Suppress INCLUDE('PRNPROP.CLW')** - This template requires that PRNPROP.CLW be included. You may already include it elsewhere, so you can suppress it here.

### 2.3.3 Export Records to CSV File using Process

#### *(Procedure Extension Template)*

This is a "quickie" export facility. It uses a regular Process procedure to export all fields within the primary file to a CSV file, like this:

```
"No", "Name"
1, "Mike"
2, "Jill"
3, "Mark"
```

It does not handle JOINed files, and you cannot selectively prevent fields from being exported. If this is insufficient, then our Super Import-Export templates will probably be a better fit for your requirements.

To use the template, just set up a regular Process procedure, and then add the mhExportBasic extension template to the procedure. The settings are straightforward:

Filename:	EXPORT.CSV
Maximum Record Size:	16,000
<input checked="" type="checkbox"/> Include field names as first record	

**Filename** - This is the desired output file. It can be a constant string or !Variable preceded by an exclamation point.

**Maximum Record Size** - This is the record length of each exported line (considering clipped field values, quotation marks, comma delimiters, etc.)

**Include field names as first record** - You may not want the fields names included in the file.

### 2.3.4 Memory Report

*(ABC: Procedure Extension Template)*

*(Clarion Legacy: Procedure Template)*

In some cases you need to print what is currently in memory. You don't want to process records, and you don't need to access files. Just print one thing and return. As you know, Clarion's reports like to process records (even if it is just one), and this can often cause problems in the calling logic.

New in Super Stuff 6.52: As we mentioned in an earlier version of the docs, Clarion addressed this deficiency in version 6. We had intended to continue to make these templates work in both C55 and C6x, but it was becoming a support nightmare (especially for the Clarion/Legacy chain, where we used a procedure template). Therefore, this template will work only with C55. Existing implementations migrated from C55 to C6x will produce a generation error (with appropriate instructions explaining how to change your approach). You will not be able to populate these templates in 6.x.

Assuming you are still in C55:

In ABC, just add the mhMemoryReport extension template to your existing report. It will prevent all record processing, and only one detail will be printed. There are no additional settings for this template.

In the case of Clarion Legacy, you must create a new procedure using the MemoryReport Procedure template. There's also an extension template to support the CPCS Report Previewer. (See the CPCS docs to learn what to put into the various settings.)

### 2.3.5 Hot Fields for Report or Process

#### *(Procedure Extension Template)*

When you create Browsers, Processes and Reports, Clarion automatically adds any hot fields that are referenced explicitly in list boxes, reports, template settings, etc. However, in situations where you are using a field in hand-coded source, and you must add that as an explicit hot field. If there are only a couple of fields, then this is not a big deal.

However, you may have situations where there are many fields that are used in your hand-coded source, and adding each of them as individual hot fields is a real hassle. In this situations, these two templates (one for Browsers, and another for Processes and Reports) will automatically add all the file's fields as hot fields.

In a Process or Report, you can just add the extension template as usual. For Browsers, however, it is a bit trickier. First you must highlight the desired browse in the Extensions window, then hit the Insert button and choose mhHotFieldsBrowse.

**One or All** - Is there just one file whose fields are to be projected, or do you want to do it for all the files listed in the tree?

**File** - If you choose "One File" above, then you must specify that file here.

**NOTE #1:** For the sake of compatibility with alternate report templates (e.g. CPCS), you are allowed to add the "Process/Report" version of this extension to any procedure. If you add it where it is incompatible, then there will be an error message during generation.

**NOTE #2:** Clarion's Legacy templates do not project dimensioned fields in Browsers, Processes and Reports. Our templates will still add the fields to the QUEUE (in the case of a Browse) and it will compile without errors, but the dimensioned fields will not be projected in the VIEW! If you happen to be using a file driver that automatically passes all fields regardless of whether they are projected, then your program will luckily work. Otherwise, the values of the dimensioned fields will not be fetched from the database. Of course, if you are using dimensioned fields, then you were already destined for hardship. <tease>

### 2.3.6 Pre-scan RecordsToProcess

#### *(Procedure Extension Template)*

Clarion's progress window does not display the actual number of records to be processed, leaving the user a bit bewildered as to how far they are through a process. This template fixes that by pre-reading the records, before the regular process proceeds. Be careful, though, because sometimes just reading the records for a process will take as long as the process itself, which could double the wait time for your user.

**Display Count as it is incremented**

**Display Count as it is incremented** - If there will be slight delay as the initial count is being done, then showing this to the user is a polite gesture.

### 2.3.7 Hide ProgressWindow

#### *(Procedure Extension Template)*

This template is used in a Report or Process to hide the ProgressWindow. It's handy if the process runs so quickly, that the window is a distraction. Alternatively, if it is a process that's called by another process/report, then perhaps you want to see only the ProgressWindow for the caller, and not the called.

One note of warning: even though the window is hidden, it still gets focus. This means that the calling window will appear to lose focus (i.e. it's caption will be dimmed), which may be confusing for your users.

There are no prompts for this template.

## 2.4 TABs

### 2.4.1 Initially select TAB other than first

#### *(Procedure Extension Template)*

Usually the first tab on a sheet structure is active when a window first opens. On some windows, however, you may prefer to have a different tab selected first. (One good example is when the first tab is a sort by "ID", but the user more frequently searches by name, which is the second tab. Visually and/or logically, you want to keep the first tab as "ID", but you want to save the user from the inconvenience of selecting the second tab every time they enter the window.)

This template does this for you. Just select the applicable sheet, followed by the desired control, and you are done.

**NOTE:** We accidentally had two copies of this template in our chain, with two different IDs, One is now extinct, as stated in its description. If the extinct version is in use in your application, you must delete it and use the current one instead. The former extension settings are still available for your convenience, but upon generation you will get an error message.

### 2.4.2 TabPopups for all SHEETS

#### *(Procedure Extension Template)*

You may have a window with a bunch of tabs in "logical" order, but when the window first opens, you want the "current" tab to be something other than the first in the designed order. This extension template does that for you.

Sheet Control:	<input type="text" value="?Sheet3"/>
Desired First Tab:	<input type="text" value="?TabC"/>
<input checked="" type="checkbox"/> Also use as first control for input focus	

**Sheet Control** - This is the sheet containing the desired tab.

**Desired First Tab** - This is the tab that you want to be selected first. Be careful, because this list contains all tabs, not just the ones for that sheet.

**Also use as first control for input focus** - When the window first opens, Clarion selects the first "focusable" control that it finds in the screen structure. If your tab leads to the fields that you prefer to get focus first, then turn this on. The default is ON.

### 2.4.3 Hide TABs During Insert

#### *(Procedure Extension Template)*

Many update forms can be very complex, with many tabs of potentially esoteric data fields, while the most crucial information is on one or two tabs. During an Insert, the user is often under pressure to get the basic data into the form (perhaps they are talking with a customer on the phone).

Alternatively, tabs may contain information like sales (or other child files), which definitely don't apply for a new record. In some cases, these tabs have update buttons that could be dangerous (programmatically) during an insert operation, and are safest for use during a subsequent update operation.

To make your users' lives easier and to save you the hassle of preventing orphaned children after an aborted insert, this template hides the tabs that are not really crucial when inserting a record. You merely specify which tabs to hide, and the template does the rest.

**NOTE:** This template requires the SaveButton control template to be present on the window. However, this is an extension template that must be added on the Extensions window. Therefore, you must highlight the "Update xxx Record on Disk" template in the Extensions list before hitting Insert to add the template. Otherwise, this template will be missing from the available list. By the way, after you add the template, it will be indented below the Update template in the extensions list, showing its dependency:



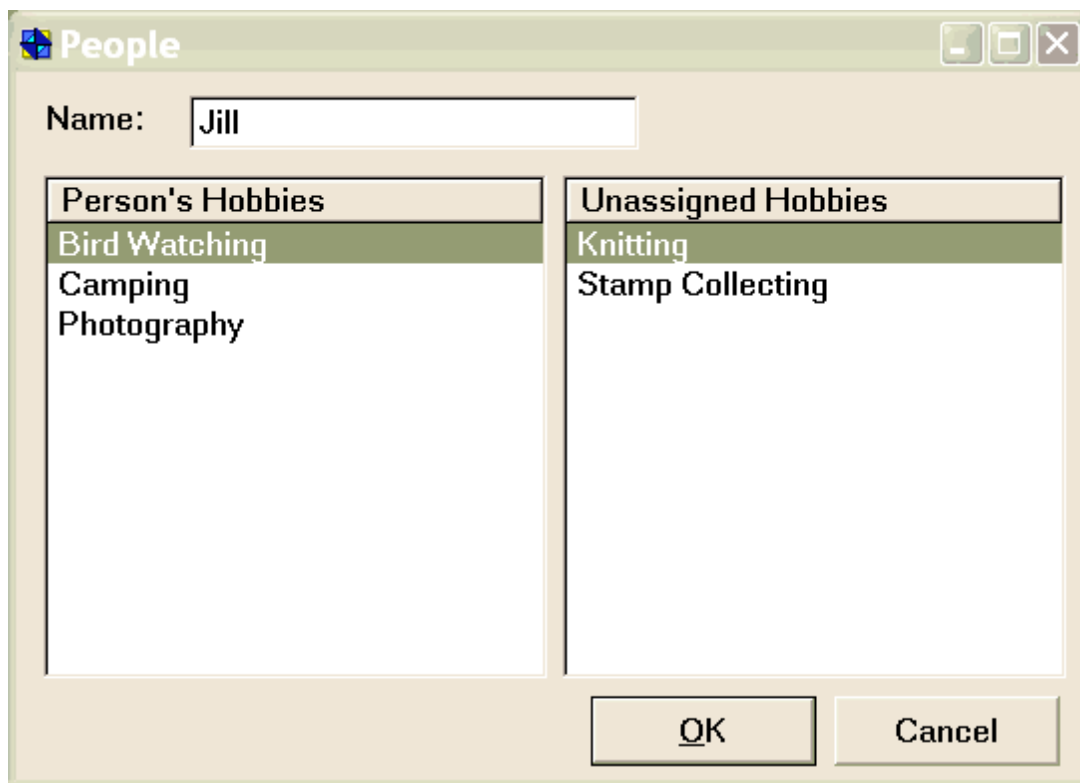
## 2.5 Enhanced Entry

### 2.5.1 Drag & Drop (MANY:MANY)

#### *(Procedure Extension Template)*

Recall that a MANY:MANY relationship is usually formed by placing a "join" file between the two files. This join file is related to each of the other files with a MANY:1 relationship.

This template enables your users to drag values between browses, assigning "attributes" to the current record. For example, you have a database of people. Each person could have zero or more hobbies. Here's a screen from the examples:



The user can drag & drop between the two browses to assign and unassign the person's hobbies. In reality there are three files involved: People, Hobby and PeopleHobby (the "join" file). When a hobby is dragged on the person's list, a PeopleHobby record is created that points to both the current person and the dragged hobby. When the user drags a hobby from the person's list back into the standard hobby list, the PeopleHobby record is deleted. The template refers to these operations as Joining and Separating.

The templates settings are very simple:

From Anchor Browse:	?ListHob	▼
To Join Browse:	?ListPeoHob	▼
Other Anchor File:	People	...
<input checked="" type="checkbox"/> Support Dragging in Both Directions		
<input checked="" type="checkbox"/> Filter "Used" from Anchor Browse		

**From Anchor Browse** - This browse contains the "catalog" of options. It represents one side of the MANY:MANY relationship.

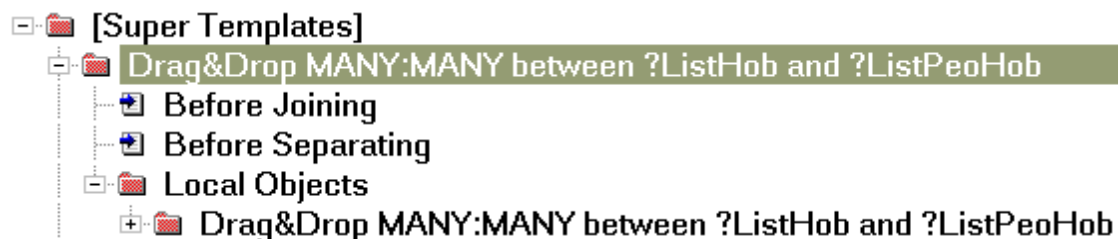
**To Join Browse** - This browse contains the join records that sit between the Anchor Browse file and the Other Anchor file. The browse must use a Range controlled by the Other Anchor file.

**Other Anchor File** - This is the other file in the MANY:MANY relationship. This file controls the range of the Join file. In our example program the People file was being updated with a form, although that's not required. All that is necessary is that the appropriate record be in memory, so that the Join record knows where it is supposed to point.

**Support Dragging in Both Directions** - In most cases you would want to be able to drag both directions, but there may be circumstances where you want to join, but not to separate.

**Filter "Used" from Anchor Browse** - Do you want already-joined entries from disappear from the list of available entries? If you are not sure, then ask yourself this question: "Can the two anchor files be joined more than once for the same pair?" If not, then the filter is probably a good idea.

## Embeds



In most situations you won't need to do anything special. Occasionally you might want to do something when Joining or Separating. If necessary, you can return LEVEL:Notify in either of these two embeds, to prevent the operation.

## 2.5.2 Enter Is Tab

*(Procedure Extension Template)*



Some users still like to use the Enter key to move from field to field, rather than "getting with the program" and learning to use the Tab key like IBM and Microsoft intended. This template keeps the luddites happy. It automatically makes the EnterKey work like a TabKey for the following control types: ENTRY, SPIN, CHECK, OPTION, COMBO, and DROPLIST.

There are two templates involved: Global and Local. Only the Global extension is required, and your entire application will be supported. If you want to tweak the operations for a particular procedure, then you simply add the Local extension to that one procedure.

### **Global Extension**

You must add this template to the global settings of all the APPs in your system. This automatically adds a global object called mhEnterIsTab. If your APP creates an EXE, then it will also add a call to initialize the object in the program setup embed.

**Enter=Tab Switch** - If some users want it others don't, you can specify a variable or expression here. It is passed with the call to mhEnterIsTab.Init, so be sure to initialize the value before that. If you need to turn it OFF or ON later, you can just execute the command mhEnterIsTab.Active=False/True. (Changing it at run-time will not affect previously opened windows.)

**Enter=Tab on BUTTONs** - Usually the EnterKey presses a button with focus. If you would rather have it press the TabKey, turn this ON.

**Add SKIP attribute to all BUTTON controls** - Some people want to press Enter to go all the way through the form. Buttons can cause problems with this, because Enter presses the button. A good solution to this is to add the SKIP attribute to the buttons, so that the Tab (which could have been an Enter) jumps over them. Recognize that this will force your users to press a button with the mouse or the button's associated hotkey.

**Also make Esc=ShiftTab** - If your users yearn for your Clarion for DOS apps, and you just can't say "No", then turn this switch ON. It will make EscKey act like ShiftTab. Be sure that all your windows have at least a button to close the window and/or the SYSTEM attribute (i.e. the [X] button in the title bar). Otherwise, you're users will be "trapped" in the window.

### **Procedure Extension**

If you want to tweak the behavior for a particular procedure, then add this extension template to that procedure.

**Suppress Enter=Tab for all controls** - If you want this procedure to be skipped entirely, then turn this ON. This also suppresses the Esc=ShiftTab feature for the entire window.

**Suppress only these controls** - If the previous setting is OFF, then you can specify individual controls here.

- For controls of the supported types, it disables the Enter=Tab feature for that one control.
- If it's a button, then it disables the SKIP option that you may have set globally.

- If the control is not one of the supported types and not a button, then including it in this list does nothing.

**NOTE:** Suppressing individual controls for Enter=Tab has no bearing on the Esc=ShiftTab feature. Esc will continue to act as ShiftTab for all controls on the window.

### 2.5.3 Quicken Date Entry

#### *(Procedure Extension Template)*

For those of you familiar with the popular Quicken accounting package, you will probably know that they have very handy date entry hot keys. Whenever you are editing a date field, you can press the following keys:

**Plus** - tomorrow  
**Minus** - yesterday  
**T** - today  
**W** - first day of the week \*  
**K** - last day of the week \*  
**M** - first day of the month \*  
**H** - last day of the month \*  
**Y** - first day of the year \*  
**R** - last day of the year \*

*\* For the keys associated with week, month and year, if already at the first/last, it goes to the next one.*

This global extension template will automatically add this feature to all the date fields in your application (i.e. any ENTRY or SPIN control with a picture of @D).

**Start of Week** - Specify whether the first day of the week will be Sunday or Monday.

Quicken Date Entry is already supported for Super Invoice and Super Browse EIP. To enable this in regular ABC Browse EIP:

1. Go into your browse procedure's Extensions list, and highlight the appropriate "Update a Record" entry.
2. Press "Configure Edit in place", change to the "Column Specific" tab, and ensure that your date column is added to the list (and is enabled, of course <g>).
3. Return to the Extensions list, and keep highlighting the "Update a Record" entry. Press Insert, then find and select mhQuickenDateRegEIP.

### 2.5.4 Limit Procedure to One Thread

#### *(Global and Procedure Extension Templates)*

Users like toolbars! Unfortunately, some users don't realize that they already have a desired

window open, so they hit the button again. The result could be countless instances of the same browse running on different threads. Not only does this chew up valuable system resources, it makes for a sloppy interface.

This template solves that problem. Start by adding the global extension template, then add this extension template to any window procedure. It will ensure that there is only one instance of the window running at a given time. If the user attempts to run the procedure again, the prior instance will be brought to the foreground. This includes restoring it from a minimized state, if necessary.

The necessary handling code is generated into the WindowManager.Run method, so that none of your Init and Kill code will be called in the case of an aborted duplicate. Also, if GlobalRequest=SelectRecord, then duplicate checking is not performed.

### **Template Settings**

**Global "Running" Thread Variable (optional)** - You may want to track the procedure's "open" status yourself, so that your program can check if it's active. If you specify a variable name here, the template will automatically set it to the thread number when the first instance starts, and then set it to zero when it stops. The best datatype for this is a SIGNED, although a BYTE is usually sufficient (because most applications don't have more than 255 threads running simultaneously).

### **Embeds**

**Single Thread Starting** - This is the first instance of the procedure.

**Single Thread Stopping** - The instance is closing (so none will be active).

**Duplicate Aborting** - A second copy has attempted to start, and is being aborted.

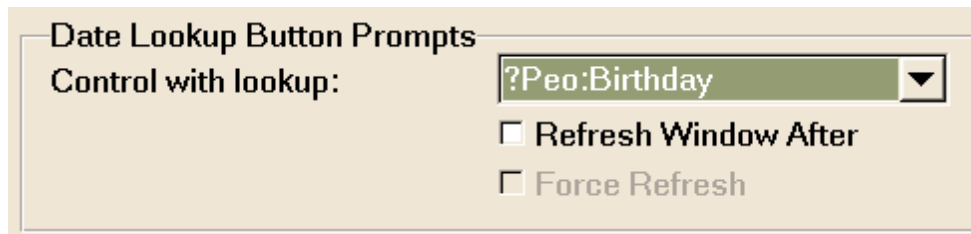
## **2.5.5 Trigger a Calendar Lookup**

### ***(Control Extension Template)***

Populate this BUTTON template beside a date field. It calls MH::Calendar for the date selection.



For this to work, you must import CALENDAR.TXA from CLARION\SUPER\SRC\_???\MHSTF\. This will add the procedure MH::Calendar to your application. If your system has multiple APPs, import this into the dictionary APP, and ensure that it is exported for use by other APPs.



**Control with lookup** - This is your date entry field.

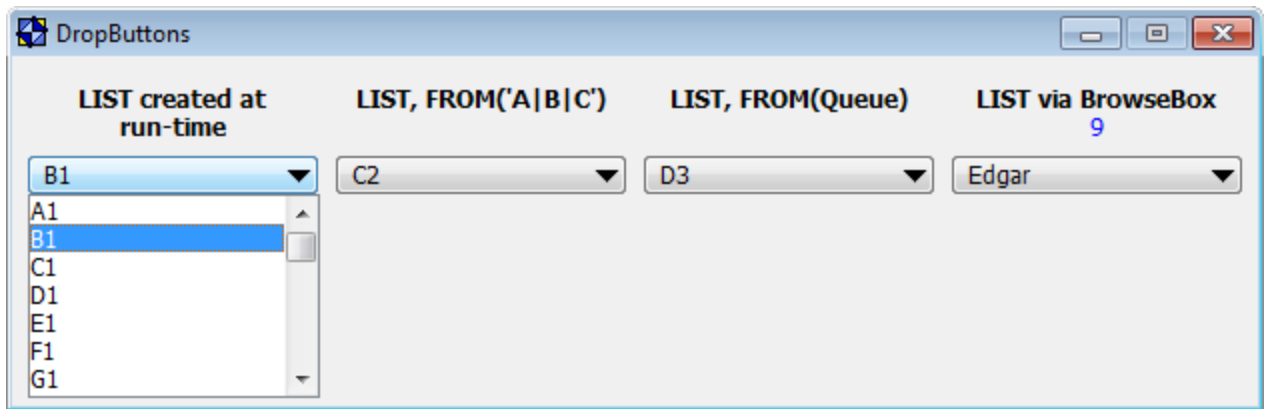
**Refresh Window After** - In ABC, this will call "ThisWindow.Reset" after a new date has been selected. For Clarion/Legacy, it will call "DO RefreshWindow".

**Force Refresh** - Do you want to force a refresh of all elements on the window?

## 2.5.6 Drop Buttons

### *(Control Extension Template)*

This control template is similar to a drop list, except that it has a much more modern look, and works with a wide range of list implementations. DropButtons are available only in ABC template chain at this time.

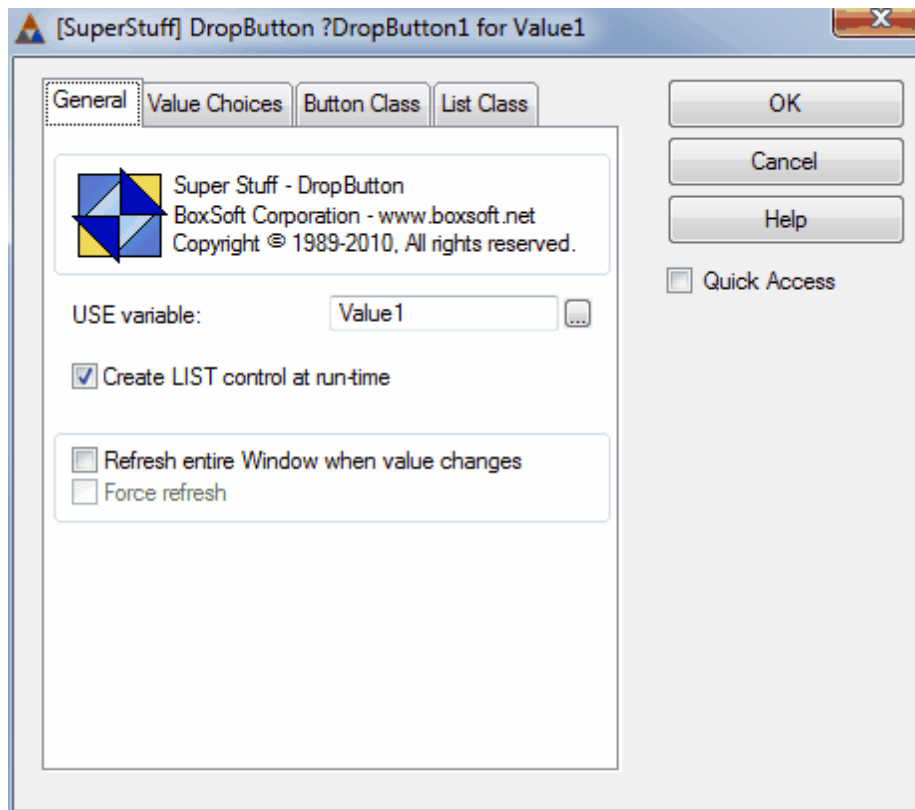


The control template is populated as a button. Then you have a variety of options to flesh it out:

- Let it create its own LIST control at runtime, in which case you must specify the values in the template settings.
- Point it at your own LIST control, which can get its values in three ways:
  - FROM('Value 1'|Value 2|Value 3')
  - FROM(SomeQueue)
  - Browse Template

In all cases, the LIST control will be hidden at runtime, until the user presses the button to "drop" the list.

## General Tab



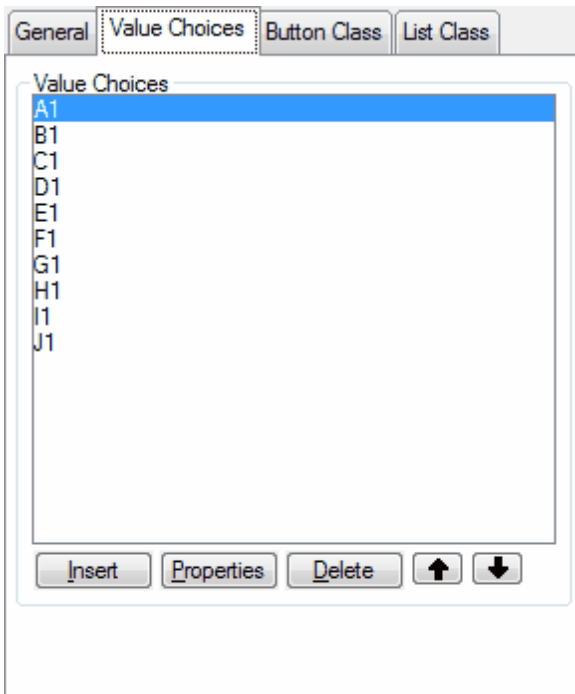
**USE Variable** - This is the variable or field that will receive the value of the choice made by the user.

**Create LIST control at run-time** - If you have a simple list that you want to define in the template settings, then turn this ON. Depending on your setting, you'll see either a "Value Choices" tab or a "LIST Control" tab.

**Refresh entire Window when value changes** - When the value changes, do you want to call ThisWindow.Reset?

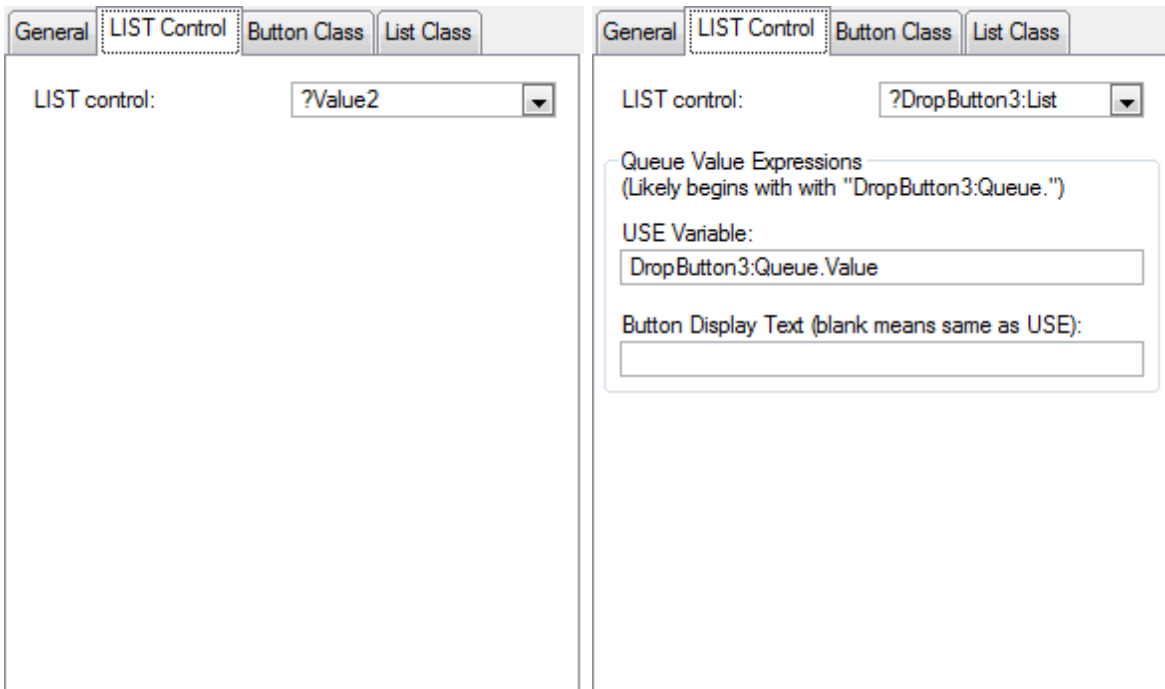
**Force Refresh** - When refreshing, do you want to force a complete refresh?

**Value Choices Tab**



This is the list of choices available in the drop list, which is created at runtime. This tab appears only if "Create LIST control at run-time" is turned ON on the General tab.

**Value Choices Tab**



You'll will see one of the three views above, depending on your LIST scenario.

- The first is FROM('A|B|C'). Note that the list's field equate is ?Value2, because its USE attribute is actually USE(Value2), rather than USE(?SomeEquate). In other words, the user's choice is automatically saved into the USE variable.
- The second is FROM(DropButton3:Queue), which is a queue that you define in your local data, and specify in the FROM attribute of your queue. You're responsible for filling this queue at run-time.
- The third uses the BrowseBox control template.

For the second and third, you must specify the queue variable (or expression) that contains the value to be assigned to your USE variable when the user makes a choice. If that value is also the thing you want the user to see, then you can leave the "Button Display Text" blank. If, however, the USE value is something like a primary key, then you can enter an expression to display something more meaningful to your user. Examples of both are shown above.

The third screen above represents a BrowseBox. The template assumes that the USE value corresponds to the browse table's primary key. If not, then you can specify an alternate key here. The key must contain only one component field.



## Button Class Tab

The screenshot shows a software configuration window with four tabs: General, Value Choices, Button Class (selected), and List Class. The Button Class tab contains the following elements:

- Class Definition** section:
- Object Name:
- Use Default ABC: MH::DropButtonClass
- Use Application Builder Class
- Base Class:
- Include File:
- Derive
- 
- 
- 
- 

This class object is responsible for managing the button, and also for communicating with the List Class object (below). You may want to rename the object to be more meaningful in your procedure. The same class is used for all list types, communicating through a shared interface called MH::IDropButtonList.

## List Class Tab

The screenshot shows a configuration window with four tabs: 'General', 'Value Choices', 'Button Class', and 'List Class'. The 'List Class' tab is active. Under the heading 'Class Definition', the 'Object Name' is set to 'DropButton1:List'. There are two checked checkboxes: 'Use Default ABC: MH::DropButtonCreateListClass' and 'Use Application Builder Class'. Below these are fields for 'Base Class' (with a dropdown arrow) and 'Include File'. A 'Derive' checkbox is unchecked. At the bottom, there are four buttons: 'New Class Methods', 'New Class Properties', 'Refresh Application Builder Class Information', and 'Application Builder Class Viewer'.

This object is responsible for managing the list. The default class will vary depending on your list style:

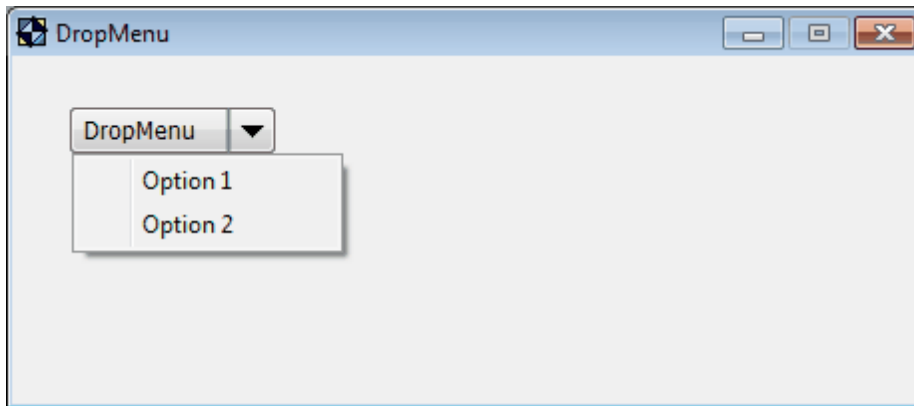
- MH::DropButton**C**reateListClass
- MH::DropButton**F**romListClass
- MH::DropButton**Q**ueueListClass
- MH::DropButton**B**rowseListClass

All of these classes implement the same interface: MH::IDropButtonList. As with the Button class, you may want to rename this object to be more meaningful in your procedure.

## 2.5.7 Drop Menus

### *(Control Extension Template)*

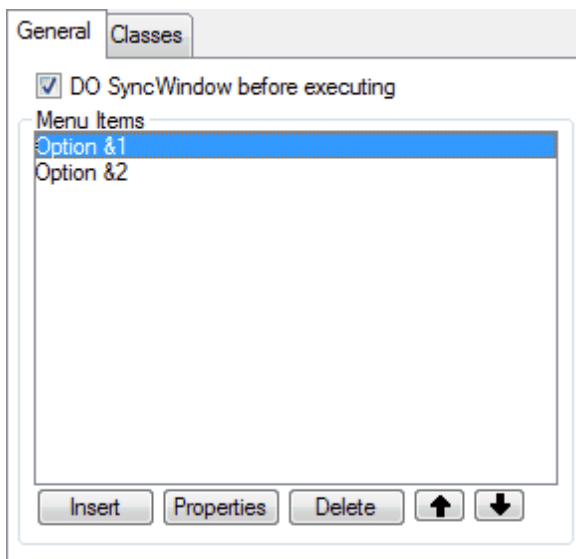
This control template uses a combination of a BUTTON and a POPUP to simulate a .NET-style drop list. Rather than selecting a data item (the job of the DropList template), you use it to access multiple actions via a single button.

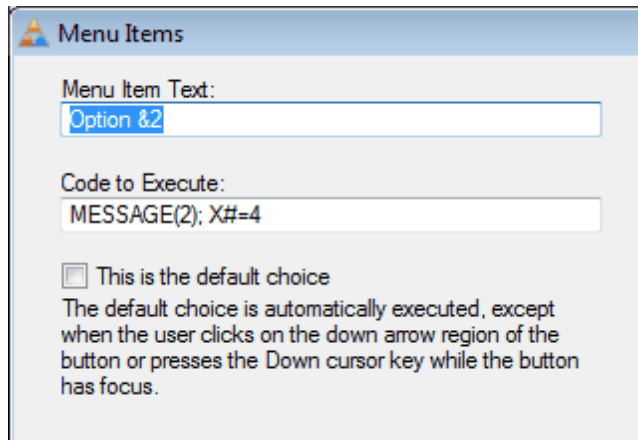


**NOTE:** If you have a multi-APP solution, be sure to add the SuperCategories global extension template to your base/dictionary DLL APP.

The control template is populated as a button, and you add your selections via the extension settings.

### **General Tab**





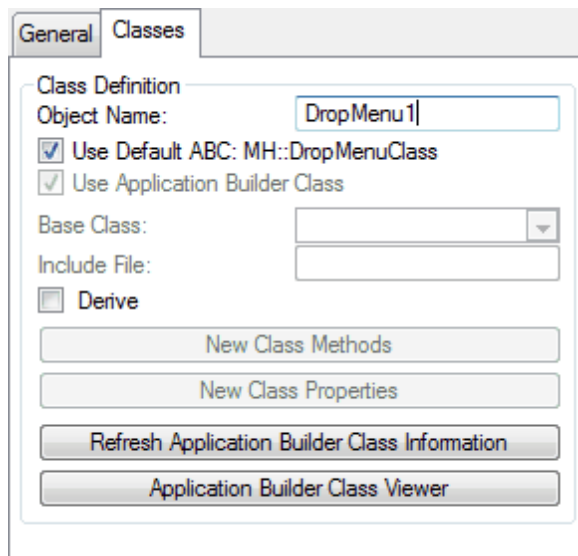
**DO SyncWindow (or Call ThisWindow.Update in ABC) before executing** - Usually you leave this ON. It causes any browse(s) to fetch the highlighted record into memory.

**Menu Item Text** - This will be displayed to the user. Use the ampersand to cause an Underlined letter. You can also type an express by beginning it with an exclamation point: !MyTextExpression.

**Code to Execute** - Enter the command(s) to perform when this option is selected. You can call a procedure, DO SomeRoutine, etc.

**This is the default choice** - If the user presses the body of the button (to the left of the button's down arrow icon), then the default option will be executed automatically, without displaying the popup. You may have only one default item.

### Classes Tab



This class object is responsible for managing the button and popup. You may want to rename the object to be more meaningful in your procedure.

## 2.6 Resize Support

### 2.6.1 Introduction to Resizing

Our approach to resizing is significantly different than Clarion's. With Clarion's resize templates, it attempts to guess how you want it to work. If this is unacceptable (which is normally the case), you must tweak each control with a rather confusing and limited array of settings. Even with that, it's expected that your windows will have one primary list box that requires resizing, with the rest of the controls being unaffected, or merely shifting about on the window.

In contrast, our resizing is entirely flexible. If you have a simple window, you can answer a couple of quick questions, and it handles the rest. If, instead, your window is very complex (e.g. 6 browses with associated buttons, with each group of controls occupying one-sixth of the window), then the mhResize support will give you the control that you need to achieve a masterful resizing effort.

1. To use mhResize, you must start with the mhResizeGlobal extension template. If you have a multi-APP project, then you must have this in all of your APPs.
2. Next comes the mhMeasureFrame extension, which is added to your main Frame procedure. It's responsible for determining how much space that you have to display windows. New in version 6.5: If your windows are all set to an initial position of CENTER, or set to Maximize Initially, then you don't need the mhMeasureFrame template.
3. Each of your window procedures can utilize the mhResizeWindow extension template to handle the bulk of the resizing operations. Finally, there's mhSplitWindow so that your users can vary the space between sections of a single window. All of these templates cooperate to create a robust and flexible resizing environment.

#### Design Standards

There are a couple of design standards to which you must adhere so that the mhResize templates can do their magic:

*New in version 6.5:* In the past we required that all your windows use the same font face and size. This is no longer necessary.

**Window Size** - Although you are not required to design with a standard window size, we have found it to be beneficial to for the windows to fit on the smallest likely client. We normally shoot for 640x480, although this may be a problem for some windows. The resizing can automatically allow any larger client to benefit from the extra space, without making life difficult for those users with smaller monitors. The mhResize templates give you the best of all worlds.

**Centering** - All windows using the mhResizeWindow template must have the CENTER attribute. You can still use the "Save/Restore Window Position" settings for the window, though. This means that the window must appear in the center only as a default. After that, the user may move it to any desired position and it will reappear with that position and size the next time it's opened.

**Resize Border** - The mhResizeWindow template can automatically add a resizable border to your windows. It's actually the default.

**Maximize Button** - For “proper” resize support, your window should have a Maximize button. Unfortunately, Clarion does not default this for you. The mhResizeWindow template can warn you (during generation time) to add this option to your window, but it cannot to it for you. (This is due to a quirk in Clarion.)

See also:

[Resize Global Template](#)

[Measure Frame for Resizing](#)

[Resize Window Template](#)

[Update Control's Position for Resize](#)

[Split Window Templates](#)


## 2.6.2 Resizing Global Support

### *(Global Extension Template)*

This template must be added to the global extensions of your app. If your system includes multiple APPs, it just be in the base APP (the one with the file definitions marked as “Internal”), as well as any other APP that contains resizable windows.

## General Tab

General Miscellaneous Classes

 Super Stuff - Resize  
BoxSoft Corporation - [www.boxsoft.net](http://www.boxsoft.net)  
Copyright © 1989-2014. All rights reserved.

Disable Resize Template throughout APP

Interpret "Maximize" event as "Resize to Full"

Make the "Full" size smaller than the frame's client area, for a visual buffer zone:

Width Adjustment:

Height Adjustment:

Tweak the median spacing used with Left- / Right- / Top- / Bottom-Half:

Median:

**Disable Resize Template throughout APP (Only ABC)** (*New in Super Stuff 6.60*) - This effectively switches off the resizing in all procedures within the current APP. You may need to do this for testing purposes.

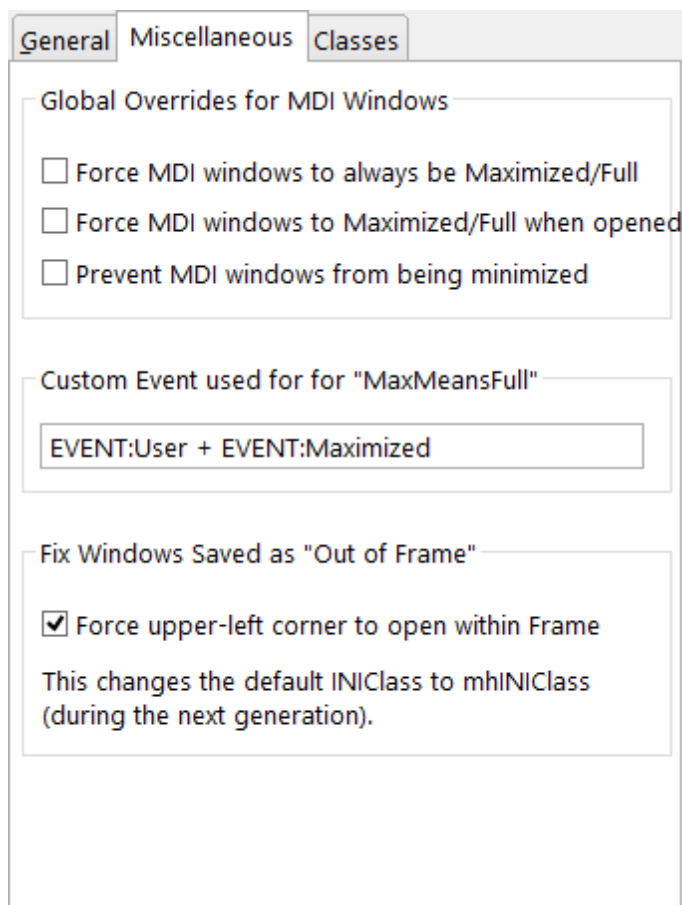
**Interpret "Maximize" event as "Resize to Full"** - Some users can get confused when the client area goes into "maximize" mode within the frame. Use this setting to make your program change this behavior. When the user hits the Maximize button on an MDI child window, it expands to the largest possible size in the frame, but doesn't actually go into maximize mode.

**"Full" size Adjustments** - If you are making use of the "Full" size for your windows, you may not want them to take all of the client area, as this looks too much like the true maximize mode. If so, use this setting to decrease the full size by a small amount, so the user see a visual buffer between the client window's borders and the limits of the frame's client area.

**Median Spacing** - This is a rather obscure setting, which you probably will never change. It controls the spacing between two independent windows. One is set to appear on the left or top half of the workspace, while the other is set to appear on the opposing (right or

bottom) half. The median is the space between the two windows. (This is not the same as Tiling, which you are more familiar with. Keep reading to make more sense of this concept.)

### **Miscellaneous Tab**



The screenshot shows a configuration window with three tabs: "General", "Miscellaneous", and "Classes". The "Miscellaneous" tab is selected. It contains three sections:

- Global Overrides for MDI Windows**: Three unchecked checkboxes:
  - Force MDI windows to always be Maximized/Full
  - Force MDI windows to Maximized/Full when opened
  - Prevent MDI windows from being minimized
- Custom Event used for for "MaxMeansFull"**: A text box containing "EVENT:User + EVENT:Maximized".
- Fix Windows Saved as "Out of Frame"**: One checked checkbox:
  - Force upper-left corner to open within Frame

This changes the default INIClass to mhINIClass (during the next generation).

**Force MDI windows to always be Maximized/Full** - This forces all MDI windows opened by your application to be resized to a large size, and prevented from changing size. This could be used to simulate a "modal" application, by reducing visual clutter.

**Force MDI windows to Maximize/Full when opened** - This initially opens MDI windows as large size, but doesn't prevent the user from changing the size.

**Prevent MDI windows from being minimized** - Users are sometimes confused when they accidentally minimize an MDI child window. This options blocks the action of the Minimize button on all child windows in your program.

**Custom Event used for "MaxMeansFull"** - This custom internal event is used to handle the initial resize operations. Hopefully it does not conflict with any other custom events that you might be using. If necessary, you can override the value here.



**Force upper-left corner to open within the Frame** - This applies if you're saving and restoring your screen sizes. Sometimes a user will move a window to the far edge of the application's client area when the frame is maximized. Later they'll run the frame in a non-maximized state, and when they open the child window, it's outside the frame area, leaving the user rather perplexed. This feature watches for that, and moves the window to a visible location when it's hidden on opening. It achieves this by replacing the regular INIClass with mhINIClass. (If you also use HandyTools, you may encounter a compiler error, due to a rogue technique that Gus Creces uses. Contact him if this occurs.)

### **Classes Tab** (New in Super Stuff 6.54)

The screenshot shows the 'Classes' tab in the application builder. It features a 'Class Definition' section with the following fields and controls:

- Object Name:** mhResizeFrame
- Use Default ABC:** MH::ResizeFrameClass
- Use Application Builder Class**
- Base Class:** AStringValue (dropdown menu)
- Include File:** (empty text field)
- Derive**
- Buttons:** New Class Methods, New Class Properties, Refresh Application Builder Class Information, Application Builder Class Viewer

Below the Class Definition section is the **Default ResizeWindow Manager** section, which includes:

- ResizeWindow Class:** MH::ResizeWindowClass

**Class Definition** - This is the global "mhResizeFrame" object. You can override it just as you would any regular template object. If you decide to change this and you are developing a multi-APP system, then be sure to use the same settings in all of the APPs.

**Default ResizeWindow Manager** - This is the default class used by all of the mhResizeWindow procedure extension templates in your APP. The default is MH::ResizeWindowClass.

See also:

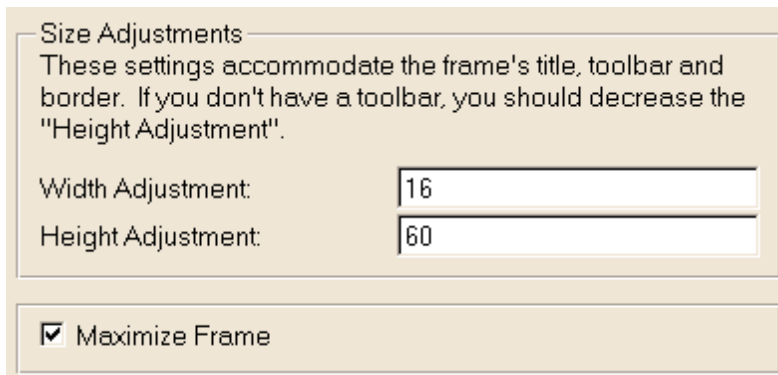
[Introduction to Resizing](#)

[Measure Frame for Resizing](#)  
[Resize Window Template](#)

### 2.6.3 Measure Frame for Resizing

#### *(Procedure Extension Template)*

This template is responsible for measuring your main Frame workspace. When subsequent windows are opened within the Frame, each of them will now know how much space is available for their positioning and sizing.



Size Adjustments  
These settings accommodate the frame's title, toolbar and border. If you don't have a toolbar, you should decrease the "Height Adjustment".

Width Adjustment:

Height Adjustment:

Maximize Frame

**Width and Height Adjustments** - mhResizeWindow has the option to open windows "Full" size.

This attempts to use the major of the Frame's workspace, without actually maximizing the window. These width and height adjustments control how much of the frame's workspace remains as a "margin" around the outside of the client windows. Depending on the size of your toolbar (or if you have one at all), the height adjustment may need to be changed. Normally the width setting is fine as-is.

**Maximize Frame** - Since we always turn on the "Initially Maximized" option for all of our Frames, we decided to have the template do it for us. You can turn this off if you don't want it.

See also:

[Introduction to Resizing](#)  
[Resize Global Template](#)  
[Resize Window Template](#)

### 2.6.4 Resize Window

#### *(Procedure Extension Template)*

This is the center point of the mhResize templates. It's the one that you place on any windows that require resizing. Unlike Clarion's WindowResize template, you cannot simply add it to procedure and walk away. You must at the very least specify the "QuickEntry" settings. As your

window becomes more complex, you can tweak it as necessary to achieve truly amazing feats of flexibility.

In addition to specifying the template settings in the Extensions window, all of your controls can also be tweaked in their Actions tabs. These will either be available directly, or via a button entitled "Super Stuff - Control Resize Settings".

**General Tab**

With most templates, the "General" tab is usually the first tab. In this case, we wanted to make the "QuickEntry" controls easy to get to, as they're the ones that would need to be changed in almost all cases. In contrast, the General settings can normally be left with their default values.

**Resize Width / Height** - These two check boxes determine whether you can resize the window in each direction. Some windows may be designed as wide as they ever need to be, so you could prevent width resizing operations. I don't think we've ever turned off the height resizing in our own development (since it's almost always good to see more items in a list control), but you can do it if you want.

**Maximize Initially** - Do you want your window to be maximized initially. This is similar to setting it in the window configuration. However, we suggest that you consider "Full" as an alternative to "Maximize". (Keep reading for more information.)

**Initial Horizontal Position** - The default for this is "Full". If you look back at the notes for the mhMeasureFrame template, you'll see a quick description of the difference between "Full" and "Maximized". Because many users get confused when dealing with an MDI database application in maximized mode, we've found that it's better to have the windows occupy "most" of the frame's workspace, yet not be maximized. (Of course,

there are always exceptions to this rule.) With this in mind, the default of “Full” means that the user can see as much information as possible, without getting confused by the maximized state. Other possible values are:

- **Center** - This will cause the window to appear in the “designed” size. That is, the size that you used to create the window. Of course, they can resize it from this starting point.
- **Left / Right** - These settings position the window on the left or right side of the frame's workspace. The initial width is the design width.
- **Left-Half / Right-Half** - These settings are similar with Left and Right, except that the initial width will be half of the frame's workspace. There is actually a “median” between two windows opened on opposing halves. This median setting is controlled in the mhResizeGlobal template.
- **Variable** (*New in Super Stuff 6.60*) - If you have a need for the window to appear in a different starting position, depending on run-time conditions, then use this setting and enter the variable name in the next field. It will expect the variable to have one of the following constant values:

```

mhHorzPos:Center      EQUATE('Center'   )
mhHorzPos:Full        EQUATE('Full'     )
mhHorzPos:Left        EQUATE('Left'     )
mhHorzPos:Right       EQUATE('Right'    )
mhHorzPos:LeftHalf    EQUATE('Left-Half' )
mhHorzPos:RightHalf   EQUATE('Right-Half')

mhVertPos:Center      EQUATE('Center'   )
mhVertPos:Full        EQUATE('Full'     )
mhVertPos:Top         EQUATE('Top'      )
mhVertPos:Bottom      EQUATE('Bottom'   )
mhVertPos:TopHalf     EQUATE('Top-Half' )
mhVertPos:BottomHalf  EQUATE('Bottom-Half')

```

**Initial Vertical Position** - This setting is similar to the Initial Horizontal Position, except that it applies to the vertical dimension.

**Extra Width / Height Adjustment** - If most of your windows use the “Full” initial positions, then the user can sometimes get lost in the consistently-placed windows. For example, we often have selection browses appear on the right-side. However, the top, right and bottom borders of that selection browse will still perfectly line up with the window that's calling it, making it seem "camouflaged". To alleviate this somewhat, you can specify these adjustments so that the window is “trimmed” in comparison to its parent. (i.e. It doesn't go all the way to the edge of the "Full" position). Start with a value of 10 and work from there.

**Original Size is Minimum Size** - In most cases, you'll design the window to be as small as possible (perhaps for a 640x480 screen). Therefore, the template assumes that the design size is the minimum size. If, however, you know that it can go smaller, then turn OFF this checkbox.

**Allow Border Only to Resize Smaller** - This switch is used for a very specific circumstance. For

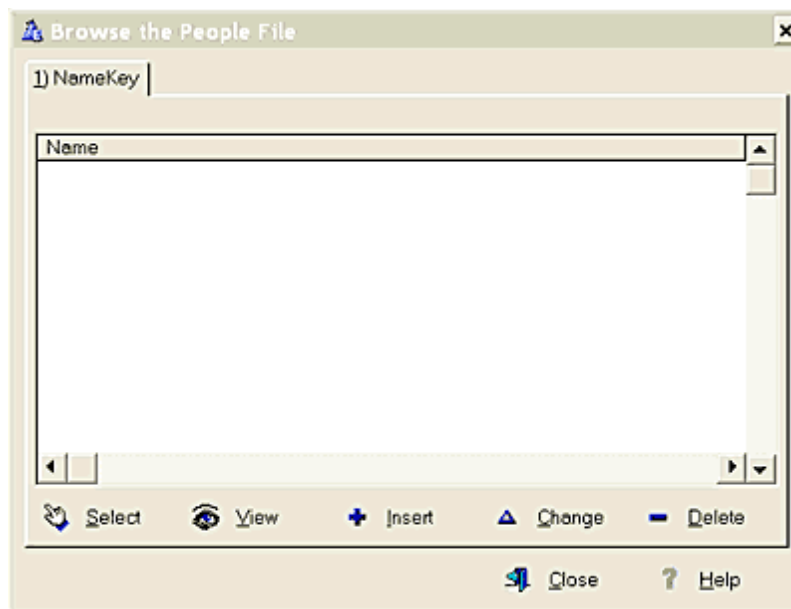
example, you have a window with two browses, one on the left and one on the right. This window is quite large, and you sometimes need to see what's under the window without moving it off the workspace. The right-side browse is not as important as the left, so you allow resizing to move the border smaller than the minimum size without resizing the contents. (Above the minimum size it will still resize the contents normally.) To get a better feel for this, try turning it on to see what it does. It might come in handy for you some day.

**Warn if Maximize Button is Missing** - Due to a quirk in Clarion it's not possible for the template to automatically add the MAX attribute to your window. All it can do is warn you if it is not there. If you actually do not want the MAX button on your window, then you can turn off this warning here.

**Add Resizable Border (if necessary)** - If your window does not have a resizable border, the template can automatically add it for you.

### **Quick Entry Tab**

This tab contains the primary settings required for a "normal" window containing some combination of a sheet, a list control, and a few controls to the right and/or below the list/sheet. Here is the window design that you'll find in the example program, in both the Browse::QuickEntry and Browse::TweakQuickEntry procedures:



The Quick Entry settings are as follows:

<b>Sheet to be Resized</b>	
<input type="checkbox"/> [RESET]	?Sheet
<b>List to be Resized</b>	
<input type="checkbox"/> [RESET]	?ListPeople
<b>Move right cut-off Control</b>	
<input type="checkbox"/> [RESET]	?Insert
	<input checked="" type="checkbox"/> Move this control too
<b>Move down cut-off Control</b>	
<input type="checkbox"/> [RESET]	?Insert
	<input checked="" type="checkbox"/> Move this control too

These controls are selected using the “FROM” template prompt type, so there's no way to unselect them once you've selected them. Consequently, we have added “Reset” check boxes to each one to allow you do clear the setting. There's actually a bug in Clarion that causes the List control setting not to clear, so you can leave the Reset switch on to maintain its cleared state.

You can actually combine these settings with individual control tweaking. The tweaking will always take precedence over the QuickEntry settings. So if you can “scoop” a bunch of controls with the QuickEntry settings, then you can deal with the stragglers with individual tweaking.

**Sheet Control** - If you happen to have a sheet that you want to resize whenever the window is resized, then select it here. The sheet is resized and not moved. It takes the entire width and height of the resize.

**List Control** - This is the single (or perhaps primary) list control that will be take the entire width and height of the resize operation (along with the Sheet, if you specified that too.) As with the sheet, it resized and not moved.

**Move Right Cut-off Control** - Often your windows will have some buttons or other controls to the right of your list box, or you have a group of controls under the list that should always be right-justified. In our sample window above, let's assume that we want the Insert, Change, Delete, Close and Help buttons to stay on the right edge of the window. You have two options to handle this:

You can specify the left-most of these controls (or if they are aligned any one will do), then tell the template to “Move this control too”. All controls with an XPos equal to or greater than the anchor control's XPos will be moved during a resize operation. (This is the method used in the above settings example.)

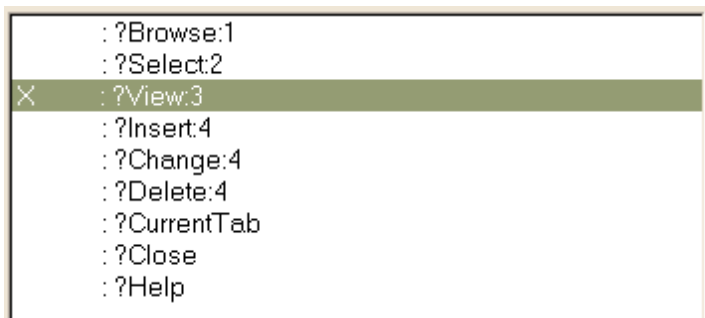
If, instead, you would rather say “everything to the right of my list box should be moved to the right”, then specify the list box as the anchor control, but turn OFF the “Move this

control too" check box.

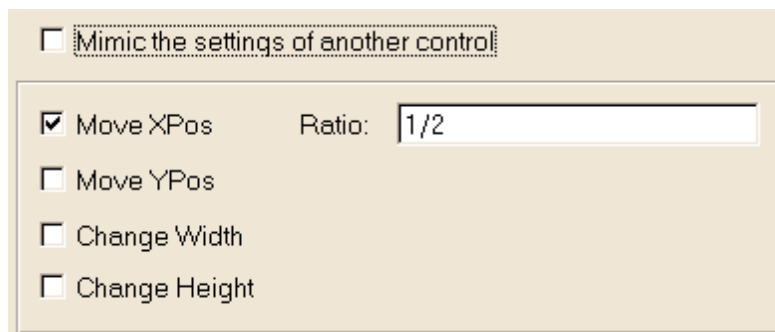
**Move Down Cut-off Control** - This is the same as the "Move Right" setting, except that it deals with controls that are to be moved down instead of to the right. (In the case of our sample settings above, we're using the List as the cut-off control, and we've turned OFF "Move this control too".)

### **Tweak Controls Tab**

This is where the real fun begins! You will see a list of all controls on your window. Any lines that are blank represent controls without an equate, and you should not set them without first assigning an equate to the control. (Also, you cannot set tweak the position of TABs, because they are controlled by their SHEET, so you won't see them here.) Here's the list for our example window above:



Note that only ?View:3 has any tweaking. This is because the QuickEntry tab has handled everything else. In the case of the View control, we want it to be centered between the Select and Insert buttons (i.e. split the difference). Therefore, we'll tweak it so that its XPos take 1/2 of the resize. The settings look like this:



**NOTE:** As was mentioned earlier, you can set some resize options in the Control settings in the window formatter. If you've done that, then you cannot adjust the control's settings here. You'll still see what they are, though, for reference purposes.

With each control, you can choose to override any or all of the XYHW ratios. These "ratios" represent the portion of any resize operation that should be applied to that dimension of the control. The default is 1. You can type a fractional amount by entering division (e.g. 1/2, 1/3, 2/3). You can also explicitly say, "do not touch this" by setting the ratio to 0 (zero). (This is necessary

on the very rare occasion, usually when your window becomes exceedingly complicated.)

There is also a "Mimic" option that can be handy when you have multiple controls doing the same thing (e.g. Insert+Change+Delete buttons). Rather than repeatedly specifying the settings for each control, you can set them for the first, and then tell the rest to mimic that control.

The easiest tweaking example is to duplicate the effects achieved with the QuickEntry tab. You've got a window with a sheet and a list. Under the list box are buttons for Select, View, Insert, Change, Delete and Close.

Start by tweaking the sheet. You don't want it to move, but you want it to be fully resized when the window is resized. Therefore, you can turn on the override for the Width and Height, and set the ratios for both to 1 (which happens to be the default). The same applies to the List control (X=off, Y=off, W=1, H=1). Or, because they are handled the same, you could have the List control mimic the Sheet (or vice versa).

In the case of the Select button, you want it to move down, but stay on the left side. Therefore, turn YPos ON, and set the Ratio to 1. Leave all other settings as-is.

The View button is tweaked as it was in the original QuickEntry example, with the addition of YPos setting for moving it down (X=1/2, Y=1, W=off, H=off).

For the Insert, Change and Delete buttons, you want them to move down and to the right. Therefore, you must override their X and Y settings to take full advantage of the resize (X=1, Y=1, W=off, H=off). At a minimum, you must do this for the Insert control. For the Change and Delete controls, however, you can either apply these same settings, or tell each of them to mimic the Insert control. The template will attempt scan backwards through the controls to find a mimicable one, which can be a real time saver.

The Close and Help buttons are set to mimic the Insert button as well. The resulting tweaking list looks like this:

```

WH : ?Browse:1
Y   : ?Select:2
XY  : ?View:3
XY  : ?Insert:4
mimic : ?Change:4 == ?Insert:4
mimic : ?Delete:4 == ?Insert:4
mimic : ?CurrentTab == ?Browse:1
mimic : ?Close == ?Insert:4
mimic : ?Help == ?Insert:4

```

A more complex example is a window with two list boxes, side-by-side. Each has its own Insert, Change and Delete buttons. There is a single Close button for the whole window. Here's what you do:

**?LeftList** must resize its height fully, but it should occupy only one half of the widow's width. It will never move from its position, so the settings will be X=off (don't move sideways), Y=off (don't move vertically), W=1/2 (resize one half of the width), and H=1 (resize all of the height).



**?RightList** is similar to ?LeftList, in that its width represents a half of the window, and its height is "full". However, it must shift to the right as ?List1 is resized. Therefore  $X=1/2$ ,  $Y=0$ ,  $W=1/2$  and  $H=1$ .

**?LeftInsert**, **?LeftChange**, and **?LeftDelete** will all have the same settings, because you want them to move as a group. (Or you can set ?LeftInsert, and have the other two mimic it.) Their size will never be changed, so only the X and Y settings will be needed. Their associated list control occupies half of the window, and you want the buttons to stay justified against the list's right edge. Also, the list uses a full-height resize, so they must move fully to stay out of the way. Therefore, the settings will be  $X=1/2$ ,  $Y=1$ . (W and H are both OFF, or you could turn them ON and set the Ratio=0.)

**?RightInsert**, **?RightChange** and **?RightDelete** are similar to their counterparts for ?LeftList. The difference is that they are hugging the right edge of ?RightList, which is always the full resize amount. (Remember that ?LeftList has  $X=1/2$  and  $W=1/2$ , totaling 1.) Therefore, their setting for  $X=1$  and  $Y=1$ . Again, you can set the options for ?RightInsert, and have ?RightChange and ?RightDelete mimic it.

**?Close** is quite simply,  $X=1$  and  $Y=1$  (i.e. put it in the bottom right corner, and don't change the button's size). Alternatively, this is the same maneuver as ?RightInsert button, so you could have ?Close mimic that.

What if you have a single list box, with three buttons beneath it? You want one button to stay justified with the left edge of the list, one control to stay justified with the right edge, and one to flow between them. This requires settings of ?Left ( $X=0$ ,  $Y=1$ ), ?Middle ( $X=1/2$ ,  $Y=1$ ), and ?Right ( $X=1$ ,  $Y=1$ ). Note that all the buttons had  $Y=1$ . Alternatively, you could have used the Move-Down Cutoff on the QuickEntry tab.

This becomes a little more complicated if you have two list boxes, ?List1 and ?List2, each with their own buttons that must be spread below them. As before, all buttons would have  $Y=1$  (or use Move-Down Cutoff in the QuickEntry tab). The value of X would be:

$?Left1=0$ ,  $?Middle1=1/2/2$  (or  $1/4$ ),  $?Right1=1/2$

$?Left2=1/2$ ,  $?Middle2=3/4$ , and  $?Right2=1$

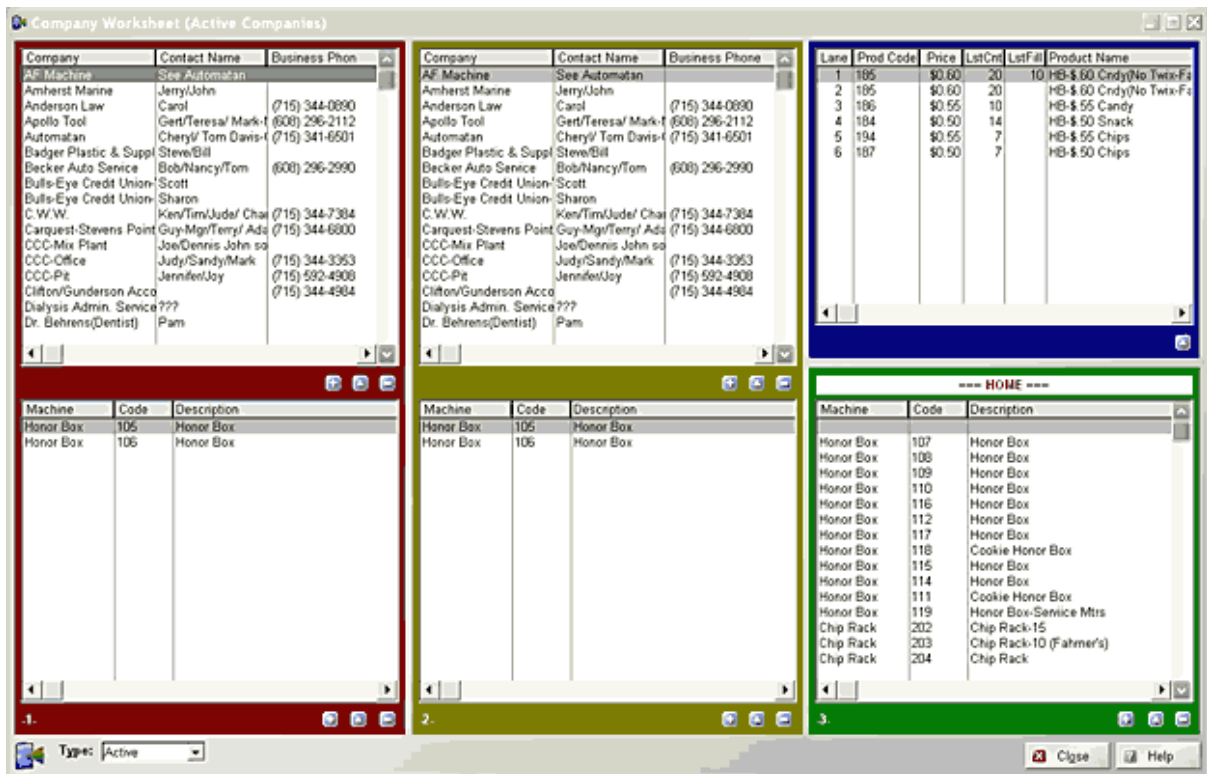
If it seems odd that ?Right1 and ?Left2 have the same "X" value, then think about it: the space between the two list boxes will be constant, therefore the space between the two buttons on the near edges will also be constant.

You can use this to create very complex resizing windows. For example, we have one window with six browses, each with 3 or more associated buttons. There are also four panels that are used to group the browses, and various additional strings and such. Of course, it took a while to specify all of settings, but at least it was possible.

The design (i.e. minimum) size looks like this:



The "Full" size on a big screen looks like this:



In this case, each browse took an equal share of the resize. We could have just as easily had the top browses taken 60%, and the bottom 40%. And the left and middle columns could have used 40% each, leaving 20% for the right column. The permutations are infinite, can be tailored to suit your exact needs, and are limited only by your imagination!

### Toolbars

If you have a toolbar on your window, then you must tell the template not to resize those controls. In other words, tweak the control, and set X, Y, W and H to 0 (zero). In some cases, you may want to have one or more buttons (i.e. ?Close) justified against the right edge of the window. To achieve this:

1. Create a local SHORT variable (e.g. CloseOffset)

2. After the window opens, set `CloseOffset = 0{PROP:Width} - ?Close{PROP:XPos}`.
3. In the `AfterResize` method, set `?Close{PROP:XPos} = 0{PROP:Width} - CloseOffset`.

We will implement a better solution in a future version.

See also:

[Introduction to Resizing](#)  
[Resize Global Template](#)  
[Measure Frame Template](#)  
[Update Control's Position for Resize](#)  
[Split Window Templates](#)

## 2.6.5 Update Control's Position for Resize

### *(Code Template)*

You have independently adjusted (via your own code) the position and/or size of a control, and you want this new position to represent the "baseline" (i.e. new "original" position). This template communicates your intent to the resizing class.

**Update Control's Position for mhResizeWindow**  
This template helps to call `mhResizeWindow's UpdateControl()` with the proper parameters.

Use it when you have changed the position or dimensions of a control using source code, and you need inform `mhResizeWindow`.

For example, after changing all dimensions, use:

```
mhResizeWindow.UpdateControl(?Control,'XYWH')
```

Control:

New XPos  
 New YPos  
 New Width  
 New Height

Just specify the control, and what your code has changed, and it will deal with it accordingly. It

automatically references the control's position and size against the current position and size of the window, relative to the window's original position and size.

See also:

[Introduction to Resizing](#)  
[Resize Window Template](#)

## 2.6.6 Split Window (Horizontal and Vertical)

### *(Control Templates)*

Sometimes you'll have a window with two browses, and you want your user to control the relative sizes of the browses. These two control templates perform this very task. If your list boxes are side-by-side, then use the `mhSplitWindowHorz` template. Otherwise use `mhSplitWindowVert`. It will place a gray region control on your window. Position this region to occupy the space between your two list boxes (adjust the height or width as necessary). Don't forget to tweak the settings for this control if you are using `mhResizeWindow` as well.

Although the splitter bar is gray on your window, it will be invisible to the user until they start moving it. If you wish to use a different color than gray, then you can change it in the region control's settings after populating the control template.

The image shows two screenshots of control templates. The top screenshot is for a horizontal split window, showing 'Left Controls' and 'Right Controls' tabs. Below the tabs, there are two input fields: 'Left Margin' and 'Right Margin', both set to the value '60'. The bottom screenshot is for a vertical split window, showing 'Top Controls' and 'Bottom Controls' tabs. Below the tabs, there are two input fields: 'Top Margin' and 'Bottom Margin', both set to the value '40'. Both screenshots have a 'Minimum / Maximum' label above the input fields.

**Left / Top Controls (`mhSplitWindowHorz`)** - Here you will specify any controls to be resized or repositioned when the splitter bar is moved. For each control you must indicate whether the control is to be resized or moved. For example, a list control would be usually be resized, while right-edge-justified controls would be moved. If your controls are left-edge-justified, then you don't need to include them here.

**Right / Bottom Controls (`mhSplitWindowVert`)** - This is similar to the Left / Top Controls, except that the options are "Resize+Move" or simply "Move". In this case, the list would be resized and moved, while any controls that are justified to the left/top would be moved (but not resized). Any controls that are justified to the right/bottom (i.e. X/YPos=1,

---

Width/Height=0) on your window can be ignored, as the normal resizing template will move them.

**Left / Top Margin** - This is the smallest allowed position of the splitter bar. If the user attempts to move the bar beyond this position, it stops.

**Right / Bottom Margin** - This represents the minimum distance from the splitter bar to the right or bottom edge of the window.

See also:

[Introduction to Resizing](#)  
[Resize Window Template](#)

## 2.7 Miscellaneous

### 2.7.1 Build Custom VIEW

#### *(Procedure Extension Template)*

Clarion's VIEW structure and associated ViewManager class represent the preferred method for accessing your data. Most of us don't want to hand code the VIEW structure, or manually instantiate the ViewManager object, along with calling its various initialization methods.

The mhView extension template makes this task easy. It presents very similar options to the regular Process template, leaving you do write a simple bit of source to access these powerful entities. This is frequently better than using an entire Process procedure. You can also have multiple mhView extensions in your procedure, while the Process procedure can access only one batch of records per procedure. Also, you can repeatedly loop through your records, and even change the filter and/or range criteria for each pass.

To make things even easier, if you want to process the view only once, you can call the "Run" method (or Routine, for legacy APPs) and it handles the opening, looping with next, and closing for you. (Don't forget to place your record handling code in the TakeRecord method.)

#### **ABC vs. Legacy/Clarion Templates**

In the ABC version, there is an object created called **Manage:ViewName**, where "ViewName" is the name you specify on the General Tab. You access the object with calls like **Manage:ViewName.Open**, **Manage:ViewName.Next()** and **Manage:ViewName.Run**. In the Legacy/Clarion version, routines are generated which perform these same options, which are named like **OpenViewName** and **NextViewName**. They perform essentially the same operations.

The ABC version contains slightly more functionality, but everything you need for the vast majority of situations is present in the Legacy/Clarion version as well. Since this template essentially provides an adjunct to hand-coded programming, there are really no roadblocks holding you back.

## General Tab

**View Name** - This will be the name of the VIEW structure itself. The ViewManager object will be called **Manage:ViewName**, where "ViewName" is the name of your VIEW. In our example above, the ViewManager object is called Manage:ViewPeople.

**Do not call mhView.Init and Kill** - The Init and Kill methods are automatically called when the files are opened and closed. (If you're doing this in a Source procedure, then you must add the mhSourceFiles extension template to your procedure.) In some situations, you may not want these to be called immediately. For example, if you want to initialize some variables after the files are opened, but before the Init method is called, then you could turn this off and call it yourself. Don't forget that you are responsible for calling both methods yourself when you turn this OFF.

**Only one record will be accessed** - Usually you will be accessing a batch of records. However, in some situations you may know that you are fetching only one record. In that situation, turn this ON and the template will tell the ViewManager object that to setup the buffers accordingly.

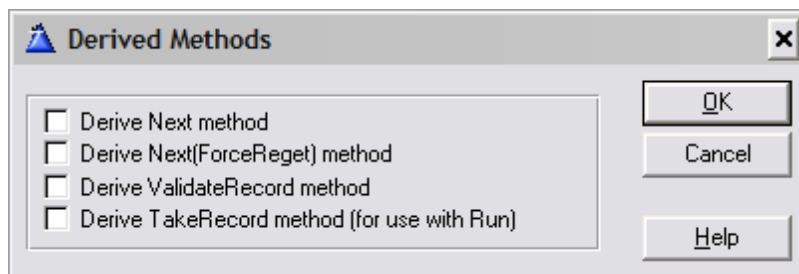
**Generate QUEUE structure and related code** - Do you want the template to generate a matching QUEUE structure for your VIEW, along with code to assign values back and forth? The queue will be called **Queue:ViewName**, where "ViewName" is the name you specify. The method to assign the record buffer fields to the queue fields is **Manage:ViewName.SetQueueRecord**, while the opposite operation is called **Manage:ViewName.UpdateBuffer**. In the Clarion/Legacy chain, the equivalent routines are

called **SetQueueRecordForViewPeople** and **UpdateBufferForViewPeople**.

**Shortcut to Files Window** - The template requires a Primary table, and it is often inconvenient to exit from the Extensions window, go to the Tables window, and then return to the Extension window. Instead, you can just press this button to jump to the table schematic window. Keep in mind that this is only a shortcut, and that the file label displayed here is not used.

**Custom Joins** - You will often have secondary files joined to your primary file. Usually these will use the relationships, as defined in your dictionary. Sometimes, however, you will need to define these differently. Here you can define settings for the secondary files, specifying a key to be used, and how the field values match up. (Of course, you must also specify that the second file uses a "Custom" join back in the table schematic.)

### Derived Methods



**Derive Next method** - Use this to derive the base ViewManager class' Next method.

**Derive Next(ForceReget) method** - Use this to derive the mhViewManager class' Next (ForceReget) method.

**Derive ValidateRecord method** - Usually you could put this type of conditional code into the processing loop itself. However, there may be situations where you would rather place this into a separate ValidateRecord method. As is usual with ValidateRecord, it should return RECORD:Ok, RECORD:Filtered or RECORD:OutOfRange.

**Derive TakeRecord method (for use with Run)** - If you intend to use the Run method rather than manually calling the Open, Next and Close methods, then you must turn this ON to derive this method and its embeds for you.

### Mini-Process Tab

This tab is available only when the template is populated into a Source procedure. If this procedure will only process these view records and exit, then you can have all of the necessary support code (open, looping, close, etc.) written for you. It also calls OpenFiles, CloseFiles, and exits the procedure when done.

For ABC, put your record handling code into the TakeRecord method. For Legacy/Clarion, use one of the "Perform action on View record" embeds.



Gen "all" code for this "single-purpose" procedure

Force Reget

Only one record will be accessed

Direction:

Return Value:

\*\*\* IMPORTANT \*\*\*

Put your record handling code in TakeRecord!

Go to Embed

**Force Reget** - If you need to REGET the view record (to align the record pointers for the projected files).

**Only one record will be accessed** - This is a display-only reminder of the setting from the General tab.

**Direction** - Do you want your view processed using Next or Previous order.

**Return Value** - If your Source procedure prototype has a return type, then you can enter an expression here, and the template will use that as the return value.

**Go to Embed** - In ABC, this button takes you to the embed tree for the TakeRecord method. In Clarion/Legacy, there are two buttons, one for before the REGET ("I've got no regets" <g>), and one for after the REGET. The REGET will be performed if you turn ON the "Force Reget" option above or place code into the "After Reget" embed.

### **Range Limits Tab**

These settings are the same as they are for any Process, Report or Browse.

### **Filter Tab**

In addition to a regular filter expression, you can also specify "Additional Field Comparisons". This is a prompted system, where you specify the field to validate (Left Field), the comparison type (Operator), and the comparison value (Right Field/Value). The comparison value can be the name of another field or variable, or a constant/expression.

**Use these before regular Record Filter** - If you specify both a regular filter expression and additional filter fields, as a default the regular expression is applied before the additional filter fields. If you want that order switched, turn this ON.

**Left Field** - This is the field whose value needs to be validated.

**Operator** - Specify the comparison type: =, <>, >, <, >=, <=.

**Select Field** - Is the Right-Hand-Side a Field (versus an value expression)? If so, then turn this ON, and use "Right Field" to select the desired field.

**Right Field** - When "Select Field" is ON, this is the field against which the Left Field is compared.

**Right Value** - When "Select Field" is OFF, this is the expression against which the Left Field is compared. If you're using the Clarion/Legacy templates, this expression will be inserted

directly into the filter expression, so all elements must be bound.

**Right Value is Constant Expression** (*ABC only*) - If you're using the ABC templates and your value expression computes to a constant value (the same value for all records), then turn this switch ON. It causes the expression to be calculated and inserted as a constant value in the filter string.

**Expression is STRING** (*ABC only*) - If the constant value specified above is a STRING, then it must be surrounded by 'quotes' in the filter expression. Turn this ON to perform this task.

### **Custom Order Tab**

Here you can control the order that the records will be processed. You can use this like the normal "Additional Order", where the custom order is applied after the key specified for the primary file. Or you can have this completely override the primary file's key. You also can specify the custom order by hand, or be prompted for each order field and its sequence.

### **Hot Fields Tab**

For our convenience, it assumes that all fields are hot. (This is opposite to regular templates, which expect you to specify each hot field individually.) If you want to optimize this, then you can turn OFF the "Make all fields hot" option, and add each hot field as you would regularly.

### **Usage Tips**

For the purpose of these examples, the View Name is assumed to be MyView, and the initial examples are for the ABC version. (There are further Legacy/Clarion tips is at the bottom.)

This method is the most precise and careful, but unnecessarily complex for most situations. Also, the Reset is automatically done by the Open, so the explicit Reset is actually redundant:

```
Manage:MyView.Open
Manage:MyView.Reset
LOOP UNTIL Manage:MyView.Next()
  DO Stuff
END
Manage:MyView.Close
```

If you have to process the loop only once, this method is acceptable. (Recognize that mhView.Reset calls Open if necessary, and the Close is called in the Kill.) You can also use this for multi-pass processing, if your range and filter settings do not change:

```
Manage:MyView.Reset
LOOP UNTIL Manage:MyView.Next()
  DO Stuff
END
```

However, in most situations the following method is preferred. mhView.Open automatically calls Reset for you, and it also reapplies any order, range and/or filter constraints that may have changed since it was last opened. This means that it will probably work for both single and multi-pass.

```

Manage:MyView.Open
LOOP UNTIL Manage:MyView.Next()
  DO Stuff
END

```

If you are doing multiple passes with changing filter and range constraints, then it is probably best if you call Close after each pass, like this:

```

LOOP RangeVar = 1 TO 10
  Manage:MyView.Open
  LOOP UNTIL Manage:MyView.Next()
    DO Stuff
  END!LOOP
  Manage:MyView.Close
END

```

In SQL, though, the reapplying does not always work as you expect. That is why you may want to set the filter manually for each pass (using either SetFilter, or preferably PROP:SQLFilter). Here are a couple of examples:

```

LOOP RangeVar = 1 TO 10
  Manage:MyView.SetFilter('MyPre:Field = '& SomeValue)
  Manage:MyView.Open
  LOOP UNTIL Manage:MyView.Next()
    DO Stuff
  END!LOOP
  Manage:MyView.Close
END

```

```

MyFile{PROP:Alias} = 'a_MyPre'
! . . .
LOOP RangeVar = 1 TO 10
  MyView{PROP:SQLFilter} = 'A.Field = '& SomeValue
  Manage:MyView.Open
  LOOP UNTIL Manage:MyView.Next()
    DO Stuff
  END
  Manage:MyView.Close
END

```

Of course, if "SomeValue" is actually a string, it will look something like this:

```

Manage:MyView.SetFilter('MyPre:Field = ''& CLIP(SomeValue) &'')

```

Or:

```

MyView{PROP:SQLFilter} = 'A.Field = ''& CLIP(SomeValue) &'')

```

Also, the SQL example assumes that your file alias is "A". This is usually the case for your primary file (unless you have overridden it with PROP:Alias). If it is not the primary table in the schematic, then you will have to look at the driver trace to determine the actual alias, or arrange to set the alias yourself.

However, these issues do not usually apply for SQL or non-SQL situations. In the vast majority of cases, you will need to do nothing specify for this to work. We mention these work-arounds only so that you will have additional options when the need arises. (By the way, the underlying causes are actually the fault of Clarion's VIEW engine and/or ViewManager class, and not specifically to our implementation of them.)

### Additional Usage Tips for Legacy/Clarion

All the range and filter initialization is done in the Open routine. The standard approach is to code it as follows:

```
DO OpenMyView
LOOP
  DO NextMyView
  IF ERRORCODE() THEN BREAK.
  DO Stuff
END
Manage:MyView.Close
```

Also, note that the Next routine does a simple NEXT(MyView), so you may just want to perform that command directly rather than calling the routine.

There's also an optional Run routine that performs the Open, Looping and Close for you, with embeds for your own record handling code. This routine is generated only if you add source to those embeds, or if you turn on the "Mini-Process" option. (If you use the Mini-Process option in a Source procedure, then this routine is called from the main code section, along with OpenFiles, CloseFiles and RETURN.)

## 2.7.2 Add Open+CloseFiles to Source Procedure

### *(Procedure Extension Template)*

Prior to Clarion 6, your Source procedures do not offer the option to open and close the used files. Rather than hand-coding these yourself, you can use the mhSourceFiles template. (This template works with prior and current versions of Clarion.)

This template generates two routines, OpenFiles and CloseFiles, which you must DO within your source code. For example:

```
SourceProc PROCEDURE
ReturnValue BYTE(LEVEL:Notify)
CODE
DO OpenFiles
!Perform the code that you did before
DO CloseFiles
RETURN ReturnValue
```

If you are using the mhView template, it will automatically call its Init and Kill methods from within these routines.

**Suppress GlobalErrors.SetProcedureName** - For ABC, the template automatically calls this method. You may wish to prevent this from happening.

**Suppress PUSHBIND+POPBIND** - The template automatically calls PUSHBIND and POPBIND, but you may not require these (usually for speed purposes, when you know that the the BIND'ing has already been handled elsewhere).

### 2.7.3 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.

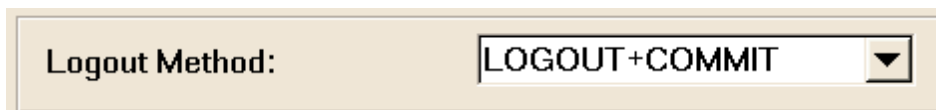
The template assumes that the window contains a SaveButton template for the Parent and a Browse for the Child. The Browse must use a "File Relationship" Range, with the Parent file as the related file.

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**

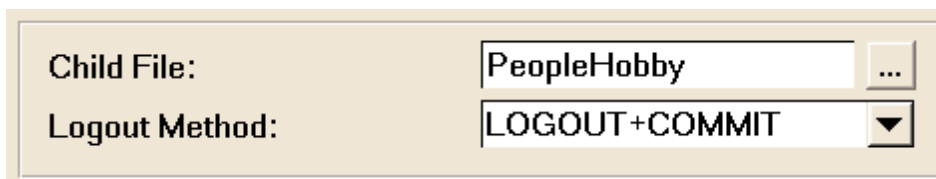
To add this template, simply go to the Extensions window from your Procedure Properties window, highlight the desired browse box, then press the [Insert] button to add the template. If you have multiple browse boxes, then you can add this extension to each one. Once you have added the template, you will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



A screenshot of a form element. On the left, the text "Logout Method:" is displayed. To its right is a dropdown menu with a white background and a black border. The dropdown menu is currently open, showing the text "LOGOUT+COMMIT" in all caps. A small black downward-pointing arrow is visible on the right side of the dropdown box.

There are no settings to specify beyond this. 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.

The non-browse version requires that you also specify the child file:



A screenshot of a form element. It contains two rows. The first row has the text "Child File:" on the left, followed by a text input field containing "PeopleHobby" and a small square button with three dots "..." on the right. The second row has the text "Logout Method:" on the left, followed by a dropdown menu with a white background and a black border. The dropdown menu is currently open, showing the text "LOGOUT+COMMIT" in all caps. A small black downward-pointing arrow is visible on the right side of the dropdown box.

If you have a child browse on the update form, we recommend that you use the first, browse-related template. For other files, the second template will work just fine, although it will not

benefit from things that the other can share with its associated browse.

### **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)
```

## **2.7.4 Look for Children records**

### ***(Procedure Extension Template)***

This template looks for child records associated with the current parent. Depending on the result, it performs some action (like changing an icon, setting a variable, etc.). This can be used in numerous situations. For example, you could have a "Notes" button on a browse that has a special icon when one or more notes are present.

### **General Tab**

<b>Files</b>	
Parent File:	<input type="text" value="People"/> ...
Child File:	<input type="text" value="PeopleHobby"/> ...
Optional Filter:	<input type="text"/>
(Remember to bind variables for the filter.)	
<b>Routines</b>	
<input checked="" type="checkbox"/> FindOne ROUTINE	<input type="text" value="MH4::FindOne"/>
<input checked="" type="checkbox"/> Process ROUTINE	<input type="text" value="MH4::Process"/>

**Parent File** - This is your parent file. (Wasn't that obvious? <grin>)

**Child File** - This is your child file. The template looks for records in this file which are related to the current record in the Parent file.

**Optional Filter** - Do you need to perform any optional filtering? If so, put a filter expression here. If you reference variables (rather than fields), be sure that they are bound.

**FindOne ROUTINE** - Do you want to generate a FindOne routine? You can override the default routine name.

**Process ROUTINE** - Do you want to generate a Process routine? You can override the default routine name.

### **FindOne Routine Tab: Action = Set Icon**

The screenshot shows a configuration window for the 'Set Icon' action. The 'Action' dropdown is set to 'Set Icon'. Under the 'Settings' section, there are three text input fields: 'Control' (empty), 'Blank Icon' (containing 'BLANK.ICO'), and 'Found Icon' (containing 'CHECK.ICO'). Below these are two event sections. The first is 'EVENT:Accepted Control', which has a checked checkbox for 'Initialize after Window is opened', a unchecked checkbox for 'Specify Accepted Control', a checked checkbox for 'Restrict to Buttons', and an empty 'Accepted Control' dropdown. The second is 'EVENT:NewSelection Control', which has a unchecked checkbox for 'Specify NewSelection Control' and a 'NewSelection Control' dropdown containing '?ListPeople'.

This is the most complex action. Essentially, you are using the template to change an icon on a button, or an an image in an IMAGE control, depending on whether children exist.

**Control** - This is the control that will host the image/icon.

**Blank Icon** - This is the icon/image that is displayed when no children exist.

**Found Icon** - This is the icon/image that is displayed when children do exist.



**Initialize after Window is opened** - Do you want this to be done when the window is first opened?

**Specify EVENT:Accepted Control** - Do you want to re-perform this check when a particular control is accepted?

**Restrict to Buttons** - Do you want the following list of available controls to include only buttons?

**Accepted Control** - This is the control to monitor for EVENT:Accepted.

**Specify EVENT:NewSelection Control** - Do you want to re-perform this check when a particular control gets a new selection (e.g. the parent browse)?

**NewSelection Control** - This is the control to monitor for EVENT:NewSelection.

#### **FindOne Routine Tab: Action = Set Variable**

The screenshot shows a settings panel for the 'FindOne Routine Tab' with the action set to 'Set Variable'. It includes a 'Settings' section with a 'Variable:' label and an empty text input field, followed by a small button with three dots.

**Variable** - This variable will be set to True, if the FindOne routine finds one. Otherwise, it will be false. You are responsible for calling the routine.

#### **FindOne Routine Tab: Action = Execute Code**

The screenshot shows a settings panel for the 'FindOne Routine Tab' with the action set to 'Execute Code'. It includes a 'Settings' section with two text input fields: '"Found" Code:' and '"Not Found" Code:'.

**"Found" and "Not Found" Code** - These code snippets will be executed appropriately. You are responsible for calling the routine.

### Process Routine Tab

The outer two are always executed.

The middle three are executed only if a child record is found.

Code	
Initial:	Count# = 0
After First:	
Activity for Each:	Count# += 1
After Last:	
Final:	MESSAGE('This person has

**Initial** - This code is executed when the process first starts, regardless of whether any children are available.

**After First** - This code is executed after the first child is found.

**Activity for Each** - This code is executed for every child that is found (including the first).

**After Last** - This code is executed after the last child is found. If only one child is found, then all three of these code snippets would be executed.

**Final** - This is executed after the process completes, regardless of whether any children were found.

## 2.7.5 Select default icon for the entire app

### *(Procedure Extension Template)*

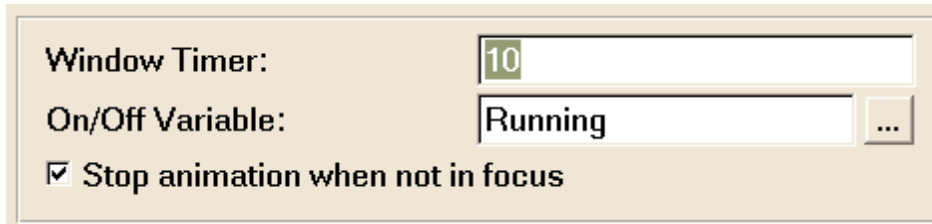
It is a hassle to remember to add icons to all windows in your system, and without them your windows cannot be minimized. This template automatically adds the specified icon to any windows that do not already have an icon of their own.

## 2.7.6 Animated Icon

### *(Control Template)*

Sometimes animated icons really tickle a user's fancy. This little template does the dirty work for you:

### General Tab



The screenshot shows a dialog box with three settings:

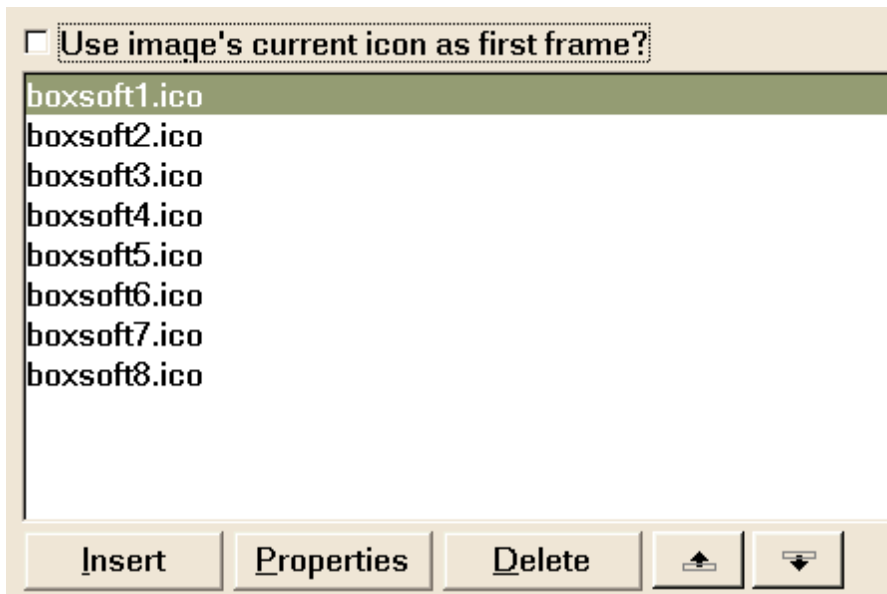
- Window Timer:** A text input field containing the number "10".
- On/Off Variable:** A dropdown menu showing "Running" and a small "..." button to the right.
- Stop animation when not in focus:** A checked checkbox.

**Window Timer** - This timer will be applied to the window. If your window already has an appropriate timer, then you can omit this one here. Remember, though, that the frame will change with each timer event.

**On/Off Variable** - If you would like to turn the animation on and off at run-time, then specify a variable here. Any non-False value means that the animation will be active.

**Stop animation when not in focus** - If this window is not the active window, then do you want to pause the animation?

### Frames Tab



The screenshot shows a dialog box with the following elements:

- Use image's current icon as first frame?:** An unchecked checkbox.
- Icon List:** A list box containing eight entries: "boxsoft1.ico", "boxsoft2.ico", "boxsoft3.ico", "boxsoft4.ico", "boxsoft5.ico", "boxsoft6.ico", "boxsoft7.ico", and "boxsoft8.ico". The first entry is highlighted.
- Buttons:** "Insert", "Properties", "Delete", and two arrow buttons (up and down).

**Use image's current icon as first frame** - If you already have an icon in your image, then you can use it as the first frame. Otherwise, it uses only those from the list below.

## 2.7.7 Prototype Exporter

### *(Procedure Extension Template)*

When your APP is intended to create a DLL, use this global template to produce a file containing the prototypes for all exportable procedures. Use this as the map include file when declaring this DLL in your other APPs.

The default name is APPNAME.INC, where "APPNAME" is the name of your application, but this may be overridden.

The output file will look something like this:

```

About                PROCEDURE,DLL(dll_mode)           !Splash and About Window
BackupHD             PROCEDURE,DLL(dll_mode)
ZCalendar            FUNCTION(<LONG>),LONG,PROC,DLL(dll_mode)
UpdateConfig         PROCEDURE,DLL(dll_mode)           !Update the Config File
LoadConfig           PROCEDURE,DLL(dll_mode)
RestoreHD            PROCEDURE,DLL(dll_mode)
BackupFD             PROCEDURE,DLL(dll_mode)
WarnBackupFD         PROCEDURE,DLL(dll_mode)
Calculator            PROCEDURE,DLL(dll_mode)
RebuildFiles         PROCEDURE,DLL(dll_mode)
RestoreFD            PROCEDURE,DLL(dll_mode)
ResetDatabase        PROCEDURE,DLL(dll_mode)
WarnErr              FUNCTION(<STRING>,<STRING>),BYTE,PROC,DLL(dll_mode)
HaltErr              PROCEDURE(<STRING>,<STRING>),DLL(dll_mode)
AutoConvert          PROCEDURE,DLL(dll_mode)

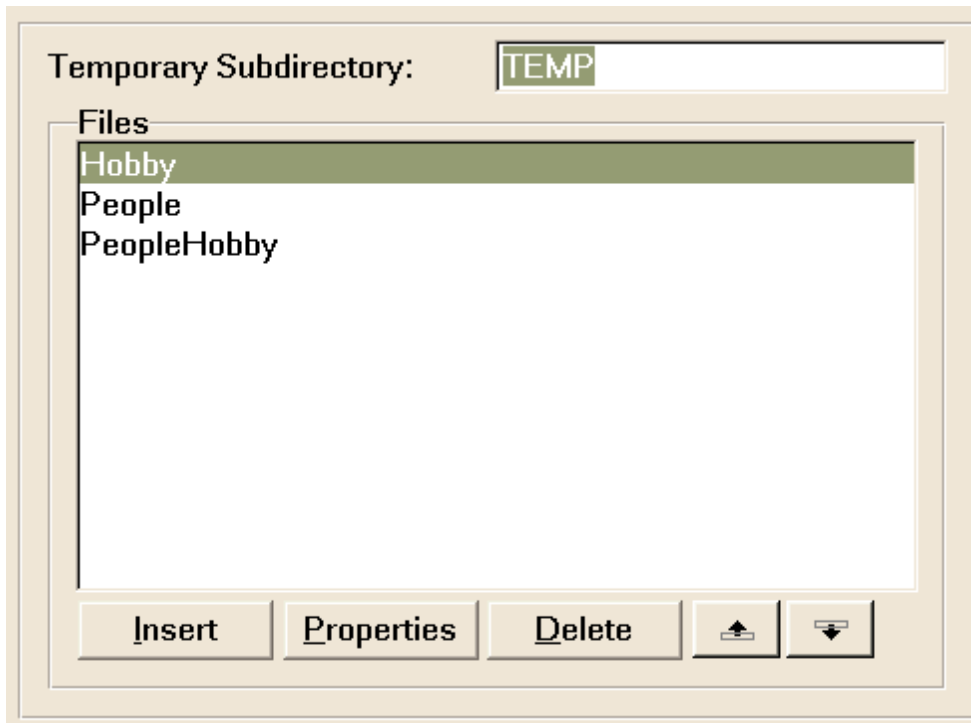
```

## 2.7.8 Make Empty \*.TPE Files

### *(Code Template)*

Clarion's TpsFix utility prompts you for example files. This code template generates empty TPE files for this purpose. This means your users can always create good example files using the current file definitions.

**NOTE:** This code template must be implemented in a Source procedure.



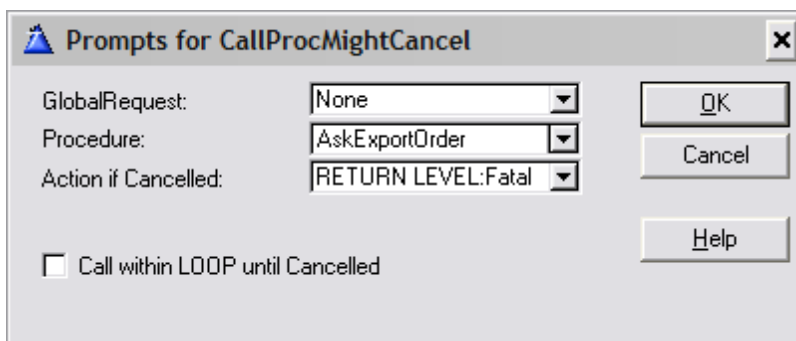
**Temporary Subdirectory** - The TPE files actually end up in the current directory, but it needs a place to create them in the first place. The subdirectory must exist before calling this, or an error will occur. It does a SETPATH to the directory before creating the files, and then a SETPATH('.') after.

**Files** - These are the files for which a TPE will be created.

### 2.7.9 Call Procedure with "Cancel" Support

*(Code Template)*

**NOTE:** This code template must be implemented in a Source procedure.



**GlobalRequest** - Do you need to set the GlobalRequest before the procedure is called? Options

are:

- **None** - not set
- **InsertRecord**
- **ChangeRecord**
- **DeleteRecord**
- **SelectRecord**
- **QbeAction** - for use with Super QBE

**Procedure** - This is the procedure that will be called.

**Action if Cancelled** - If the procedure returns with `GlobalResponse=RequestCancelled`, then this action will be performed. The options vary, depending on whether you are using the ABC or Clarion template chain:

- **RETURN LEVEL:Fatal** (*Only ABC*) - This is your most like choice, and is used when calling procedures from `ThisWindow.Init`. You may want to insert the code template in an embed before or after the files are opened, depending on your circumstances. In almost all instances, it should be done before the window is opened.
- **RETURN**
- **RETURN RequestCancelled** (*Only ABC*)
- **DO ProcedureReturn**
- **POST(EVENT:CloseWindow)** - This is used when cancellation indicates that the current window should be closed.
- **ThisWindow.SetResponse(RequestCancelled)** (*Only ABC*) - This is similar to the above option, except that it uses the `WindowManager's SetResponse` method both to post the event, and to set the local Response to `RequestCancelled`.
- **CYCLE**
- **Cancel Browse Insert** - This setting assumes that you have a Browse Control on the window, and that you want to use this code template to simulate the Insert button. As such, the `GlobalRequest` setting must be "InsertRecord", and you must specify the Browse Control to use. If you choose this option, the "Call within LOOP..." setting will be unavailable.
- **Custom Code**

**Call within LOOP until Cancelled** - Do you want to call this procedure repeatedly, until the user cancels?

**Browse Control** - If your action is "Cancel Browse Insert", then you must select the browse control here.

**Exit at end** (*Only Clarion/Legacy*) - If your action is "Cancel Browse Insert", then you can optionally choose to EXIT, if you've chosen an embed within a ROUTINE.

## 2.7.10 Embed Point Guidance

### *(Procedure Extension Template)*

This template adds a variety of described embeds throughout your procedure, to help you to insert your code in the "best" location. For example:

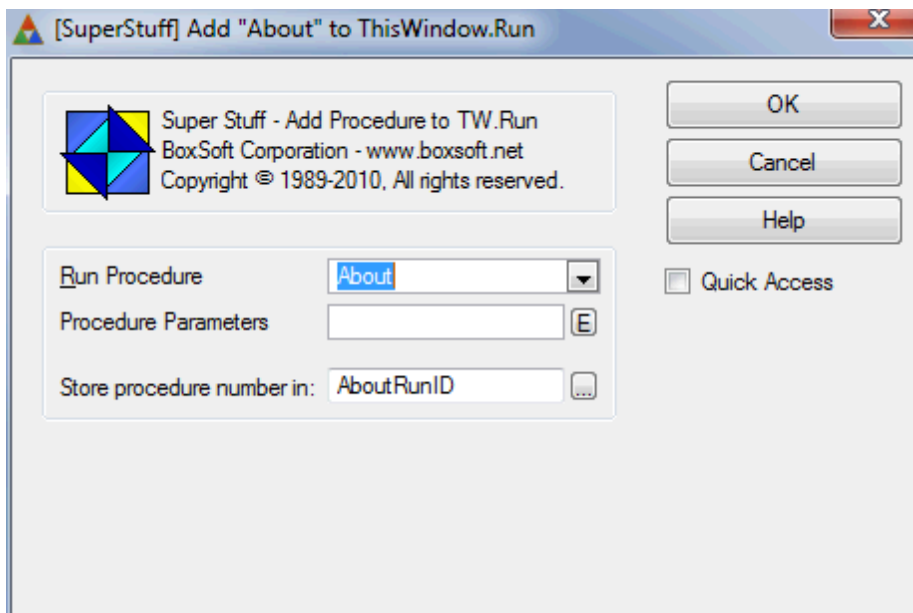
- Top of Procedure Data Section
- EQUATE Definitions
- TYPE Definitions
- PROCEDURE Declarations local to Procedure (i.e. procedure's local MAP)
- Bottom of Procedure Data Section
- PROCEDURE Definitions local to Procedure (i.e. the code for the procedures defined in the local MAP)

More will be likely added in the future, as their usefulness comes to us.

## 2.7.11 Add Procedure to ThisWindow.Run

### *(Procedure Extension Template)*

There are situations where you need to add your procedure to the derived ThisWindow.Run procedure. This is done automatically for things like browse update forms, but there's no way to manually add a procedure to that list. This templates does that for you.



**Run Procedure** - This is the procedure that you need to call.

**Procedure Parameters** - If your procedure takes any parameters, specify them here. If these parameters can change at runtime, then you must use local variables here, and set their

values before calling ThisWindow.Run.

**Store procedure number in...** - As you change the functionality of your procedure, the list of procedures handled by ThisWindow.Run may change. Therefore, you cannot call ThisWindow.Run with a hard-coded number. Specify a local variable here, and the template will automatically assign the appropriate Run ID in ThisWindow.Init. Use that variable in your call to ThisWindow.Run.

For example, ThisWindow.Run may look like this:

```
ThisWindow.Run PROCEDURE(USHORT Number,BYTE Request)
ReturnValu     BYTE,AUTO
CODE
ReturnValue = PARENT.Run(Number,Request)
IF SELF.Request = ViewRecord
    ReturnValue = RequestCancelled           ! Always return RequestCancelled
if the form was opened in ViewRecord mode
ELSE
    GlobalRequest = Request
    EXECUTE Number
        About
        BrowsePeople
    END
    ReturnValue = GlobalResponse
END
RETURN ReturnValue
```

Call the procedure like this:

```
YourResponse = ThisWindow.Run(AboutRunID, YourRequest)
```

Depending on your situation, your procedure may or may not care about the Request and Response.



## 3 Appendices

### 3.1 Examples

There are two big example programs, one each in C:\CLARION6\SUPER\EX\_ABC\STUFF (for ABC) and C:\CLARION6\SUPER\EX\_CLA\STUFF (for Clarion Legacy).

Rather than create individual procedures to demonstrate the features singly, we have heavily overloaded the functionality on a few key procedures. This means that although the frame's menu lists most of the features, many of these menu items lead to the same place. In most cases, this is either BrowsePeople or UpdatePeople. In each situation, we try to take you directly to the place where the feature is highlighted. If you take a look at how we have created each procedure, all should come clear.

You can find the various templates used in the following locations:

#### **Browsets**

##### [Threaded Browse+Form](#) (Only ABC)

- Global Extension
- BrowsePeople
- UpdatePeople

##### [Wrap Browse Field](#) (Only ABC)

- BrowsePeople

##### [Toggle "Active" Field in Browse](#)

- BrowsePeople

##### [Select/Cancel Support](#)

- Global Extension
- BrowsePeople (with Select button)

##### [EntryLocators respond to EnterKey](#)

- BrowsePeople

##### Hot Fields for Browse

- BrowsePeople

#### **Reports/Process**

##### [Ask Date Range for Report](#)

- DateRangeReport

##### [Use a Non-Default Printer for the Report](#)

- DateRangeReport

##### [Export Records to CSV File using Process](#)

- ExportToCSV

#### Hot Fields for Report or Process

- ExportToCSV
- DateRangeReport

#### Pre-scan RecordsToProcess *(Only Clarion/Legacy)*

- DateRangeReport

#### Hide ProgressWindow

- ExportToCSV

### **TABs**

#### Initially select TAB other than first

- BrowsePeople

#### TabPopups for all SHEETs *(Only ABC)*

- Global Extension (setup)
- TabPopups (demonstrated)
- BrowsePeople (demonstrated)
- UpdatePeople (demonstrated)

#### Hide TABs During Insert

- UpdatePeople

### **Enhanced Entry**

#### Drag & Drop (MANY:MANY) *(Only ABC)*

- UpdatePeople

#### Enter Is Tab *(Only ABC)*

- Global Extension

#### Quicken Date Entry

- Global Extension (setup)
- QuickenDate (see it in action)

#### Limit Procedure to One Thread

- BrowsePeople

#### Trigger a Calendar Lookup

- MH::Calendar
- UpdatePeople
- AskDateRange

### **Resize Support**

#### Resizing Global Support

- Global Extensions

---

### [Measure Frame for Resizing](#)

- Main

### [Resize Window](#)

- BrowsePeople (Quick Entry, but note Tweaking of X=0 to prevent movement)
- UpdatePeople (Tweak Controls; note in ABC, we tweak it in the window formatter actions)

### [Split Window \(Horizontal and Vertical\) \(Only ABC\)](#)

- UpdatePeople

## **Miscellaneous**

### [Build Custom VIEW](#)

- HandCodedVIEW
- Call4Change:UpdatePeople

### [Add Open+CloseFiles to Source Procedure](#)

- HandCodedVIEW
- Call4Select:BrowsePeople
- Call4Change:UpdatePeople
- Call4Insert:UpdatePeople

### [Look for Children records](#)

- BrowsePeople

### [Select default icon for the entire app](#)

- Global Extension

### [Animated Icon \(Only Clarion/Legacy\)](#)

- Animation (use the button to turn it on and off)

### [Make Empty \\*.TPE Files \(Only Clarion/Legacy\)](#)

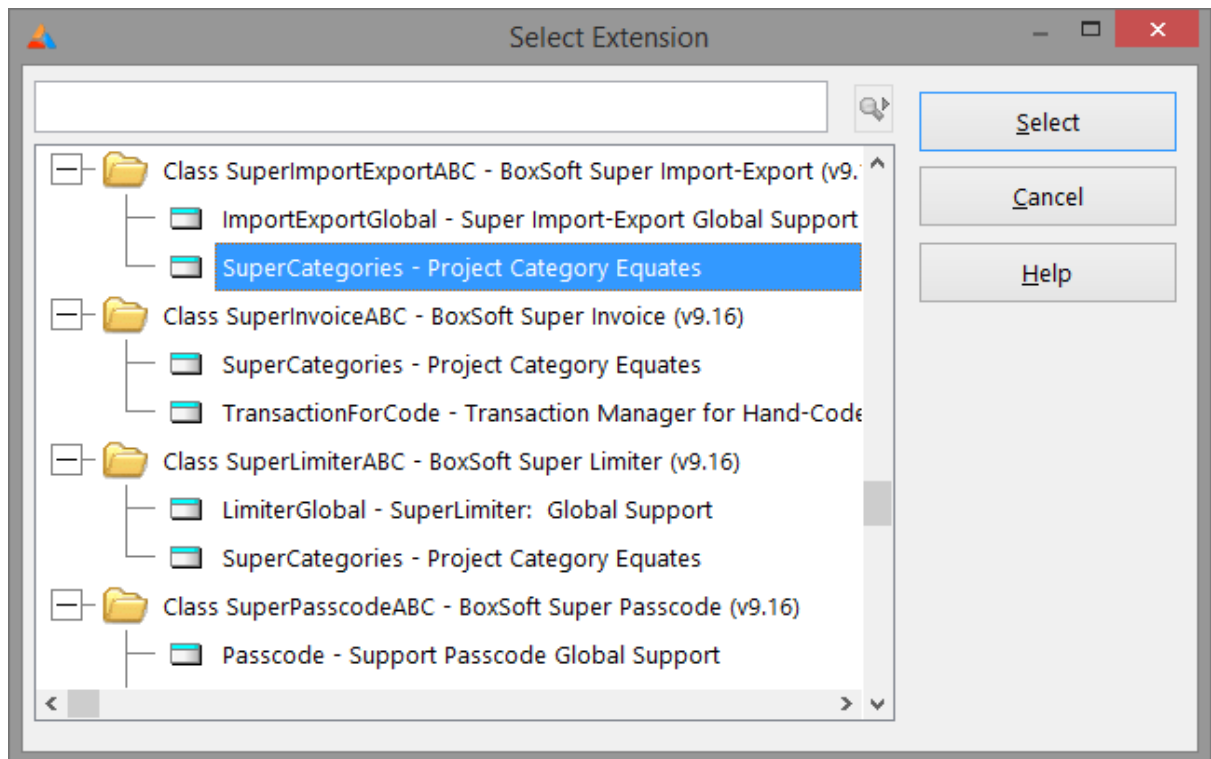
- MakeTPE

## 3.2 Project Defines

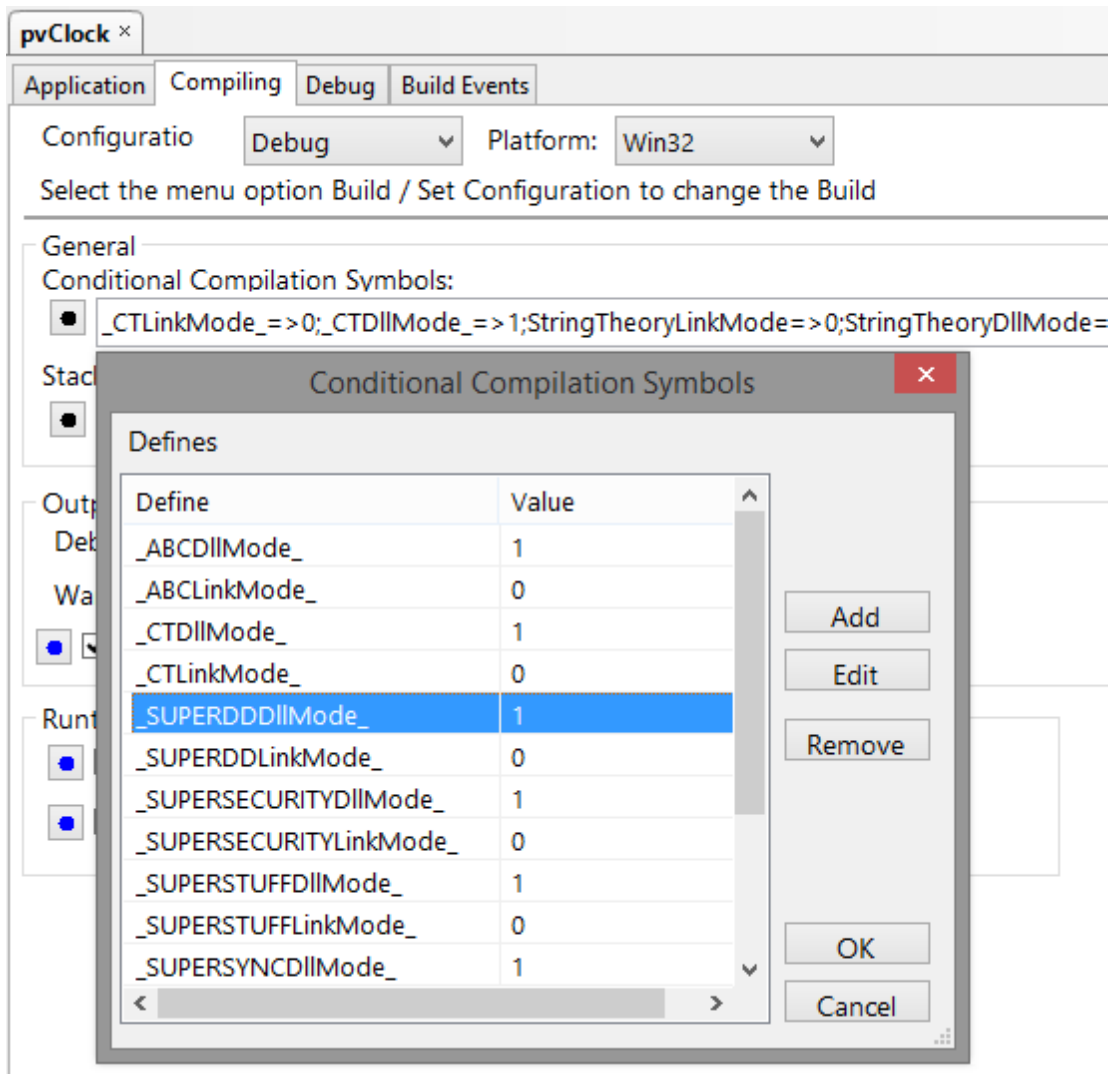
Prior to our 7.0 template versions, we were utilizing the same LINK and DLL Project Defines as the ABC libraries: `_ABCLinkMode_` and `_ABCDIIMode_`. This caused all of our libraries to be included in your base/dictionary DLL, even if you weren't using them in a particular development project.

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:

You have installed version 6.5 or later over a pre-6.5 version, and you receive error messages regarding groups missing for files named STABMH\*.TPW or STABCL\*.TPW. Or you see errors related to missing StandardAddIconToProject, and sometimes unresolved externals for MH\* or ST\* classes.

#### Solution:

You probably have a mix of versions. With version 6.5 we renamed our filenames to conform to the 8.3 standard, and you likely have a few old files kicking around. For example, here are some old and new filenames:

OLD	NEW
STAB_MHSTF.TPL	STAM_STF.TPL
STABMHR SZ.TPW	STAMRSZ.TPW
STABMHR SZ.INC	STAMRSZ.INC

When you installed the templates, did you specify the destination was the root Clarion directory (e.g. C:\CLARION6), or did you use something else (e.g. C:\CLARION6\3RDPARTY)? If you did not use the root directory, then you must manually copy ST?M\*.INC from your destination LIBSRC directory into Clarion's LIBSRC directory, and delete ST??MH\*.INC from Clarion's LIBSRC directory.

Search for all occurrences of ST??\_MHSTF.TPL and ST??MH\*.TPW. None should exist. Also look for ST??MH\*.INC and CLW. Again, none should exist.

Finally, do a search for "#TEMPLATE(MikeHanson" in \*.TPL. You should find only STAM\_STF.TPL and STCM\_STF.TPL.

#### Problem:

Your window has a toolbar, and the window is not resizing properly.

#### Solution:

See [Resize Window](#).

#### Problem:

You have a multi-APP system. When you compile one of the non-dictionary/base apps, it complains about unresolved externals that mention Super Stuff feature. This will be similar, although not necessarily identical, to this:

```
Unresolved External DESTRUCT@F28SSTF::LIMITSTARTSGLOBALCLASS in FTM_06.obj - C:\_FTM for
FEW\FTM\FTM_06.MAP:1
Unresolved External VMT$SSTF::LIMITSTARTSGLOBALCLASS in FTM_06.obj - C:\_FTM for
FEW\FTM\map\debug\FTM_06.MAP:2
Unresolved External CONSTRUCT@F28SSTF::LIMITSTARTSGLOBALCLASS in FTM_06.obj - C:\_FTM for
FEW\FTM\map\debug\FTM_06.MAP:3
Unresolved External GETISREDUNDANT@F27SSTF::LIMITSTARTSLOCALCLASS in FTM_06012.obj - C:\_FTM
for FEW\FTM\map\debug\FTM_06.MAP:4
```

---

```
Unresolved External GETISACTIVE@F27SSTF::LIMITSTARTSLOCALCLASS in FTM_06012.obj - C:\_FTM
for FEW\FTM\map\debug\FTM_06.MAP:5
Unresolved External CHECKOUT@F28SSTF::LIMITSTARTSGLOBALCLASS in FTM_06012.obj - C:\_FTM for
FEW\FTM\map\debug\FTM_06.MAP:6
Unresolved External CHECKIN@F28SSTF::LIMITSTARTSGLOBALCLASS27SSTF::LIMITSTARTSLOCALCLASS in
FTM_06012.obj - C:\_FTM for FEW\FTM\map\debug\FTM_06.MAP:7
```

**Solution:**

See Project Defines.

### 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](http://www.mittensoftware.com)  
[answers@mittensoftware.com](mailto:answers@mittensoftware.com)  
[www.boxsoft.net](http://www.boxsoft.net)  
[www.boxsoft.net/contact.htm](http://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 -

Active 16  
 Adding SuperStuff to your Applications 11  
 Alert 32  
 Animation 74  
 Application 74  
 ASCII 25

## - B -

Border 50, 60  
 Break 15  
 Browse 12, 15, 16, 36  
 Button 21, 35, 36, 42

## - C -

Calendar 24, 34, 35  
 Call 77, 79  
 Cancel 70, 77, 79  
 Cancel Button 21  
 Child 31, 70  
 Children 71  
 Class Libraries 7, 84  
 Close 69  
 Close Button 21  
 CloseFiles 69  
 Contacting Technical Support 88  
 Control 32, 35, 36, 42, 50, 59, 60  
 CSV 25  
 Current Record 26

## - D -

Date 24, 34, 35  
 Day 34  
 Deactivate 16  
 Default Printer 25  
 Defines 84  
 Directories 7  
 Disappear 84

DLL 76  
 DLLMode 84  
 Drag 31  
 Drag & Drop 31  
 Drop 31

## - E -

End Date 24  
 Enter 32  
 EnterKey 22  
 EntryLocator 22  
 Examples 81  
 Execute 71  
 EXP 76  
 Export 25, 76

## - F -

File 22, 27  
 Files 62, 69  
 Filter 16, 24, 62  
 Find One 71  
 First Tab 29  
 Form 12, 30  
 Frame 45, 46, 50, 74  
 Freeze 84

## - G -

Global 12, 21, 32, 76  
 GPF 84

## - H -

Height 50, 59, 60  
 Hide 28  
 Horizontal 50, 60  
 Hot Field 22, 27

## - I -

Icon 71, 74  
 Image 74  
 Inactive 16  
 Include Files 7

Initial Tab 29  
Insert 30  
Installation 7  
Introduction 4  
Invoice 70

## - L -

License Agreement 89  
Limit 34  
LinkMode 84  
List 36  
Local 32  
Look 71  
Lookup 35, 36

## - M -

Many 31  
Many:MANY 31  
MDI 34  
Measure 45, 46, 50  
Memo 15  
Memory 26  
Menu 42  
Month 34  
Move 50, 59, 60  
Multi-Line 15

## - O -

One Record 26  
One Thread 34  
Open 29, 69  
OpenFiles 69  
Order 62

## - P -

Parent 31, 71  
Popup 29, 42  
Pre-Scan 27  
Printer 25  
Procedure 27, 76  
Process 22, 24, 25, 27, 28, 71  
Program 74  
ProgressWindow 27, 28

Project Defines 84  
Prototype 76

## - Q -

Quick Entry 50

## - R -

Range 24, 62  
Reactivate 16  
RecordsToProcess 27  
Redirection File 7  
Registering the Template 7  
Relationship 31  
Repair 76  
Report 22, 24, 25, 26, 27, 28  
Resize 45, 46, 50, 59, 60  
Restore Child After Cancel 70  
Rows 15  
RTFM Warning 6  
Run 79

## - S -

Scroll 29  
Select 29  
Select Button 21  
Sheet 29, 30  
Size 50, 59, 60  
Source 62, 69  
Split 50, 60  
Start 34  
Start Date 24  
Support 88

## - T -

Tab 29, 30, 32  
TabKey 22  
Tech Support 88  
Template Registry 7  
TEST.APP 81  
TEST.DCT 81  
Text 15  
ThisWindow 79  
Thread 12, 34

Threaded 12  
Today 34  
TPE 76  
TPS 76  
TpsFix 76  
Transaction 70  
Troubleshooting 86  
Tweak Controls 50

## - V -

Variable 71  
Vertical 50, 60  
View 69

## - W -

Week 34  
Width 50, 59, 60  
Window 45, 46, 50, 59, 60, 74  
Wrap 15

## - X -

XPos 50

## - Y -

Year 34  
YPos 50