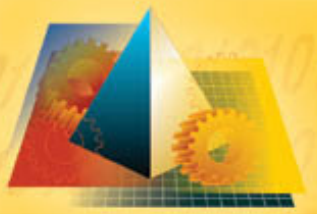


Clarion

Template Guide

SoftVelocity Incorporated

2335 East Atlantic Blvd. Suite 410

Pompano Beach, Florida 33062

(954) 785-4555

www.softvelocity.com

Documentation for Clarion 7

Printed in the United States of America (0409)

Table Of Contents

Template Guide 1

General Topics 1
Request and Response 1
Refresh Window routine 2
Application Template 3
Select Application Template 3
Select Module Template 3

Extension Templates 4

Select Extension Template 4
Additional Sort Fields Dialog 6
Add ADO File Extension 8
ADO Login Extension Template 8
ADO Support Template - Global Extension 10
ADO Process Extension 14
Data Columns/Hot Fields 16
Default SQL 17
SQL Tab Prompts 19
BLOBInControl Extension Template 21
CleanCloseDown Global Extension 22
Date Time Display extension template 23
DbAuditing 24
ExtendProgressWindow 25
FormVcrControls extension template 26
FileUserTags 26
FrameExtension Template 27
Global Cooperative Threading Extension 29
MDI Synchronization - Global Extension 30
MDI Synchronization - Procedure Extension 31
Preemptive Procedure Extension 31
Process Transaction Frame extension template 32
Record Validation extension template 33
ReportChildFiles 34
RunCommandLineProc 35
Save Button Transaction Frame extension template 35
Set DLL Image Base Memory Address 36
Version Resource extension template 37
WindowResize extension template 40
Wise-Generate Wise Installation Script 45
Business Rules Manager - Global Extension Settings 46
Global Rules 48
Rules - Controls 50
Local Business Rules Manager 51
Local Business Rules 53
Local Business Rules - Hot Fields 53
Local Business Rules - Locally Defined 54
Local Business Rules - Special Controls 55
Local Business Rules - Used Controls 55
Local Business Rules - Used Global Rules 56

Code Templates 57

Select Embed Type 57
ADO Browse Refresh Code Template 59
ADO Browse to XML Code Template 59

CallABCMethod ... 59
Call a Procedure (Extended) code template ... 60
Call Procedure As Lookup code template ... 62
Close Current Window code template ... 62
Control Value Validation code template ... 62
DisplayPopupMenu ... 63
Export to XML code template ... 65
FromXML code template ... 66
GetFullDragSetting code template ... 67
GetOSVersion code template ... 67
Import From XML code template ... 68
InitiateThread code template ... 69
Lookup Non-Related Record code template ... 70
Process Transaction Frame Checkpoint code template ... 70
ResizeSetStrategy ... 71
SelectToolbarTarget code template ... 72
SetABCProperty ... 72
SetFullDragSetting code template ... 73
SetProperty code template ... 73
ToXML code template ... 74
ViewXML code template ... 75

Control Templates 76

Select Control Template ... 76
ADO Browse Box Control Template ... 80
ADO Browse Box Select Button ... 88
ADO Browse Process Button ... 88
ADO Browse Update Button control template ... 89
ADO Errors Object List Control Template ... 90
ADO Process Pause Button ... 90
ADO Save Button Control Template ... 91
ADOLoginControls Control Template ... 93
ASCII Print Button control template ... 93
ASCII Search Button control template ... 93
ASCII View control template ... 94
ASCII View in List box ... 95
Browse Box control template ... 96
BrowseFuzzyMatching control template ... 107
BrowseGrid ... 109
BrowsePrintButton ... 110
BrowsePublishButton (BrowseToHTML) ... 111
BrowseNoRecords Button control template ... 112
BrowseQBEList Control Template ... 113
Browse Select button control template ... 116
BrowseToolboxButton ... 116
BrowseToolbarControl ... 116
BrowseUpdateButtons control template ... 117
BrowseViewButton ... 121
Calendar Button Control Template ... 121
    Calendar Button - General Tab ... 122
    Calendar Button - Classes Tab ... 123
Cancel Button control template ... 125
Close Button control template ... 125
DOS File Lookup control template ... 126
DynamicImage ... 128
Field Lookup Button control template ... 128
File Drop control template ... 129

File Drop Combo control template 135
FormVCRButtons Control Template 140
FrameBrowseControl 143
Help Button control template 145
IPDRV Options 145
MultiChildRelationTree control template 146
MultiChildRelationTree Button control template 154
MultiChildRelationTree Expand/Contract Buttons control template 155
MultiChildRelationTree Select Button control template 155
MultiChildRelationTree Update Buttons control template 156
OLE control template 158
Pause Button Control template 159
ProcessReportQBEButton 160
BrowseQueryButton 163
RelationTree control template 166
Relation Tree Update Buttons control template 169
Relation Tree Expand/Contract Buttons control template 169
ReportDateStamp control template 169
ReportPageNumber control template 169
ReportTimeStamp control template 170
RTFStatusBar Control Template 170
RTFTextControl Control Template 171
RTFToolBar Control Template 175
Save Button control template 178
SaveChangeButton Control Template 180
Sort Order Button control template 180
Sort Order Drop List control template 181

Procedure Templates 182

Select Procedure Template or Defaults 182
ADO Browse Procedure 184
ADO Form 184
ADO Login Procedure Template 184
ADO Process Procedure 184
ADO Report Procedure 184

Wizard and Utility Templates 185

Select Utility Dialog 185
Application Wizard utility template 186
Wizard--Control model 189
Browse Procedure Wizard utility template 190
Form Wizard utility template 192
Label Wizard utility template 195
ProcedureCallTreeReport Utility Template 197
Process Wizard utility template 198
Report Wizard utility template 199
Window Wizard utility template 201
Write Help IDs (WriteHLPIDs) Utility Template 202
Dictionary Print Wizard utility template 203

Wizard Themes 204

Browse Customization 204
Button Customization 206
Control Customization 207
Form Customization 208
Frame Customization 210
Process Customization 213

Process Control Customization ... 214
Report Customization ... 215
Report Label Customization ... 220
Report Additions Customization ... 223
Save As Theme ... 224
Save Theme ... 224
Sort Order Customization ... 225
Theme Design ... 226
Theme Maintenance Wizard ... 228
Window Customization ... 229

Additional Libraries and Templates 231

Crystal Reports Templates 232

Crystal Reports Overview ... 232
Crystal Reports Code Templates ... 234
GetCrystalFormulaPreview / GetCrystalFormulaPrint ... 235
GetCrystalQueryPreview / GetCrystalQueryPrint ... 235
PreviewCrystalReport ... 236
PrintCrystalReport ... 239
SetCrystalFormulaPreview / SetCrystalFormulaPrint ... 240
SetCrystalQueryPreview / SetCrystalQueryPrint ... 241
Crystal8 Class ... 242
Crystal8 Class Properties ... 242
Crystal8 Methods ... 243
AllowPrompt (prompt for runtime parameter data) ... 244
CanDrillDown(allow Crystal drill down support) ... 244
HasCancelButton (display cancel button on report preview) ... 245
HasCloseButton (display close button on report preview) ... 245
HasExportButton (display export button on report preview) ... 246
HasLaunchButton (display launch button on report preview) ... 246
HasNavigationControls (display navigation controls on report preview) ... 247
HasPrintButton (display print button on report preview) ... 247
HasPrintSetupButton (display print setup button on report preview) ... 248
HasProgressControls (display progress controls on report preview) ... 248
HasRefreshButton (display refresh button on report preview) ... 249
HasSearchButton (display search button on report preview) ... 249
HasZoomControl (display zoom control on report preview) ... 250
Init (initialize Crystal8 object) ... 250
Kill (shut down Crystal8 object) ... 250
Preview (preview a Crystal Report) ... 251
_Print (print a Crystal Report) ... 252
Query (retrieve or set the SQL data query) ... 252
SelectionFormula (retrieve or set the Crystal formula) ... 253
ShowDocumentTips (show tips on docuement in the preview window) ... 253
ShowReportControls (show print controls) ... 254
ShowToolbarTips (show tips on preview window toolbar) ... 254

Finance Library 255

AMORTIZE (amortize loan for specific number of payments) ... 256
APR (annual percentage rate) ... 258
COMPINT (compound interest) ... 259
CONTINT (continuous compounding interest) ... 261
DAYS360 (days difference based on 360-day year) ... 262
FV (future value) ... 263
IRR (internal rate of return) ... 264
NPV (net present value) ... 266
PERS (periods of annuity) ... 268

PMT (payment of annuity) ... 270
PREPMT (payment of annuity with prepayment) ... 272
PREFV (future value with prepayment) ... 273
PREPERS (periods of annuity with prepayment) ... 274
PRERATE (rate of annuity with prepayment) ... 276
PREPV (present value with prepayment) ... 278
PV (present value) ... 279
RATE (rate of annuity) ... 280
SIMPINT (simple interest) ... 282

HTML Help Templates 283

HTML Help - Overview ... 283
HTML Help Templates ... 284
cwHHGlobal extension template ... 285
cwHHProc extension template ... 286
AlinkLookUp code template ... 287
CloseHelp code template ... 288
GetHelpFileName code template ... 288
GetTopicName code template ... 288
KeyWordLookUp code template ... 289
SetHelpFileName code template ... 289
SetTopicName code template ... 289
ShowIndex code template ... 290
ShowSearch code template ... 290
ShowTOC code template ... 290
ShowTopic code template ... 290

MenuStyleManager Class 291

MenuStyleManager Properties ... 292
MenuFEQ(Menu Field Equate) ... 292
MenuStyleManager Methods ... 293
AlignShortcutsToLeft (position menu shortcuts to left) ... 294
AreShortcutsLeftAligned (detect left aligned menu shortcuts) ... 294
GetEndColor (get ending gradient color) ... 295
GetFlatMode(get menu flat mode) ... 296
GetFont(get menu text font) ... 296
GetImage (get image name assigned to menu part) ... 297
GetIsVerticalGradient (does menu part use a vertical gradient) ... 298
GetStartColor (get starting gradient color) ... 299
GetTextColor(get menu text color) ... 300
Init (initialize the MenuStyleManager object) ... 301
InitMenuInterface (initialize menu properties) ... 302
MenuHasGradient (does menu part have gradient applied) ... 303
MenuInterface (IMenuInterface reference) ... 304
MenuStyle (IMenuStyle reference) ... 305
MixColors (mix two colors) ... 306
SetColor (specify color for menu part) ... 307
SetFlatMode (set menu flat mode) ... 309
SetFont (set menu text font) ... 310
SetImage (apply image to menu part) ... 311
SetTextColor (set menu text color) ... 312
SetThemeColors (set colors based on theme detected) ... 313
SetVerticalGradient (set vertical color gradient) ... 314

Statistics Library 315

FACTORIAL (factorial of a number) ... 316
FREQUENCY (frequency count of an item in a set) ... 317

LOWERQUARTILE (lower quartile value of a set) 318
MEAN (mean of a set) 319
MEDIAN (median of a set) 320
MIDRANGE (midrange of a set) 321
MODE (mode of a set) 322
PERCENTILE (pth percentile value of a set) 323
RANGEOFSET (range of a set) 325
RVALUE (linear regression correlation coefficient) 326
SS (sum of squares) 327
SSXY (sum of squares for x and y) 328
ST1 (student's t for a single mean) 329
SDEVIATIONP (standard deviation of a population) 330
SDEVIATIONS (standard deviation of a sample) 331
SUMM (summation of a set) 332
UPPERQUARTILE (upper quartile value of a set) 333
VARIANCEP (variance of a population) 334
VARIANCES (variance of a sample) 335

Index **336**

Template Guide

General Topics

Request and Response

One of the biggest considerations of template design is inter-procedure communication. The added dimension of multi-threading only serves to make this more complex.

In a generic template-driven system, it is impossible to require that parameters be supported in templates. It's never certain if a Browse will be calling a Form, or if it calls a Report, etc. In fact, with Control Templates, a form can also *be* a browse, and an ASCII viewer. To require users to know all of the different parameters and their values is unreasonable. Further, building in support for functions would overcomplicate the templates to the point of unusability. Again, add in the complications of multithreading and the system is unusable and difficult to maintain.

Using global variables is acceptable, with the THREAD attribute ensuring that the variable itself is safe within a thread. Unfortunately, the value of any global variable must be called into question as soon as any EMBED point is encountered, as the value could change with another procedure call, etc.

Any communications variable must therefore have as little happen from the time it is assigned a value and the time that value is interpreted. This time is referred to as the *Span* of the variable. The shorter span, the better the integrity of the system. The communications variable should also be considered suspect as soon as possible. The amount of time that the variable is considered to have a valid value is referred to as *Live Time*. If a variable has a short *Live Time*, its less likely to be subject to misinterpretation, and again system integrity benefits.

To this end, we've implemented a *Request and Response* system in the Clarion templates. This system was created to maintain the integrity of interprocedure communications in a fully generated system. In other words, if no embedded source is used and no hand-coded modules are used, confidence in system integrity is high.

There are three components to the Request and Response system:

Global Variables: GlobalRequest and GlobalResponse. In GENERATED code, there are no points between the place that either variable is assigned a value and the place that that value is interpreted. These variables are defined as:

GlobalRequest LONG,THREAD

GlobalResponse LONG,THREAD

If you are creating an application that consists of more than one AppGen created DLL, you MUST check the "Generate Internal Global Data as EXTERNAL" check box for all DLLs except one. Likewise, you MUST check the "Generate Internal Global Data as EXTERNAL" check box for each APP creating an .EXE.

Local Variables: ThisWindow.Request (or SELF.Request), ThisWindow.Response (or SELF.Response), ThisWindow.OriginalRequest (or SELF.OriginalRequest).

ThisWindow.Request (or SELF.Request) and ThisWindow.OriginalRequest (or SELF.OriginalRequest) are assigned value immediately after the procedure begins.

ThisWindow.Response (or SELF.Response) is assigned a value before a bit of code signals the exit of the procedure. Right before the procedure terminates, GlobalResponse is assigned the value of ThisWindow.Response (or SELF.Response).

Enumerated EQUATEs: These are primarily to increase readability of the code. The actual numbers themselves are inconsequential, with one exception; Request values less than 0 are reserved for use in multi-page systems.

```
InsertRecord        EQUATE (1)      ! Add a record to table
ChangeRecord        EQUATE (2)      ! Change the current record
DeleteRecord        EQUATE (3)      ! Delete the current record
SelectRecord        EQUATE (4)      ! Select the current record
ProcessRecord       EQUATE (5)      ! Process the current record
ViewRecord          EQUATE (6)      ! View the current record
SaveRecord          EQUATE (7)      ! Save the current record

RequestCompleted    EQUATE (1)      ! Update Completed
RequestCancelled    EQUATE (2)      ! Update Aborted
```

Refresh Window routine

The Refresh Window routine is used by the templates to keep the values displayed current. This routine is called whenever a control's value is modified, or in the case of a list box, a different record is highlighted. The routine reevaluates the display conditions for all window controls, and displays the correct value.

Application Template

Select Application Template

Located in the Application Properties dialog

If you've added third party or your own templates to the template registry, and they include Application templates, this lets you choose which template set controls source code generation.

CLICK on an item from the list, then press the **Select** button.

Select Module Template

Clarion provides the following Module templates:

Highlight an item from the list, then press the **Select** button.

GENERATED Source	Use this choice when the module to add is source code which is generated by the Application Generator. This allows you to control the name of the Module.
External DLL	External Dynamic Link Module--Use this choice when the module to add is a DLL.
External Lib	External Library Module--Use this choice when the module to add is a .LIB (without a .DLL).
External OBJ	External Object Module--Use this choice when the module to add is a .OBJ file.
External Source	Use this choice when the module to add is Source Code that will not be maintained by the Application Generator. The Application Generator reads and compiles this source file but does not write to it.

Extension Templates

Select Extension Template

Extension templates add functionality to procedures, but are not bound to a particular control or embed point. Each Extension template has one well-defined task. For example, the Date Time Display lets you display the date and a running clock. If you add third party, or your own customized templates to the Template Registry, they appear in the list.

From a **Procedure Properties** dialog, add an Extension template by pressing the **Extensions** button. CLICK on an extension template from the list, then press the **Select** button. Here are some of the Extension templates that Clarion provides:

ASCII View in List Box	This extension provides a LIST control to alternate its display between a selected file and some other data that you specify.
BLOB in Control	The BLOBInControlExt extension template allows you to display and/or edit a BLOB from a selected table.
cwRTFGlobal	This global extension enables the use of an RTF control in the application. It includes all necessary EQUATEs and Classes. **Deprecated and replaced** with the RTF Text Control template.
Date Time Display	This extension adds a "live" date and/or time (updated every second) display to the procedure
DBAuditing	The DbAuditing template is a global extension used to track updates to specific data tables. This extension is added to the global properties of an application.
FileUserTags	This global template extension identifies certain tables and columns that possess a particular **User Options** tag defined in the dictionary.
Record Validation	This extension enables enforcement of dictionary-defined field value validation
RunCommandLineProc	The RunCommandLineProc template provides a simple way to check for command line parameters when the program is run. Several command line parameters may be defined and sorted in the order they should be checked at runtime.
FormVcrControls	This Extension template adds functionality to a Form procedure by enabling navigation and field history with the *FrameBrowseControl* VCR buttons. See *FrameBrowseControl* for more information on these buttons and their operation.
ViewFormActions	This Extension template adds functionality to a Form procedure by enabling it to open in View Only mode. You specify what mode to set for each control.
WindowResize	This Extension template lets the end user resize windows that have traditionally been fixed in size due to the controls they contain (List boxes, entry controls, buttons, etc.).

Wise-Generate Wise Installation Script	This Extension template automatically generates a Wise Installation script for your application.

Additional Sort Fields Dialog

The Additional Sort Fields Dialog is a built-in extension of the Browse, Process, and Report procedures. Its function is to add additional sort fields to the default sequence that data is processed. This sequence can be applied to the procedure's default sort behavior, or to conditional behavior if supported. If no default sort is specified (no key), the additional sort fields dialog sets the primary sort.

This dialog is supported in both ABC and Clarion template chains. The sort sequence selected applies to the appropriate ABC method or ORDER property required.

For Clarion Chain and SQL users:

For the re-positioning of the list box cursor against a SQL back end and using the Clarion Template Chain, the columns defined in the Additional Sort Fields do NOT come into play. You MUST define a KEY in the dictionary that has the Columns that you need in order to define a Unique Row in the database. You will need such a Key defined for all Sort Orders that you require. Additionally, if you have an Index on the back end that corresponds to this key, the Database Engine will use it, thereby enhancing the performance for that particular order.

Sort Type

Manual

Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Additional Sort Fields** prompt. This allows you to manually construct a sort string using the following syntax:
"+*FieldLabelOne,-FieldLabelTwo,…***"**

An incorrect string will result in no sorting, and may generate a compile error.

Assisted

The Assisted method is designed to build a correct sort string. Use the update buttons provided to add your sort fields to the list. In the additional dialog, provide the following information:

Field

Use the ellipsis to select a field name from the *Select Column* dialog.

Order

Select Ascending or Descending from the drop list, to designate an ascending or descending sort order.

ConditionalAssisted

Valid for Report and Process only. The Conditional Assisted sort type adds a condition to the assisted method, where the additional sort fields will only be applied when the condition is true.

Condition

Use the ellipsis to select a field name from the *Select Column* dialog. When this field or column is non-zero, the additional sort fields will be applied.

Field

Use the ellipsis to select a field name from the *Select Column* dialog.

Order

Select Ascending or Descending from the drop list, to designate an ascending or descending sort order.

DynamicSorting

Valid for Report and Process only. This Sort Type generates a dynamic sort window at runtime, where your users can modify (prior to generating the report) the sort components as to the report's order and component sort type (ascending or descending). Use the update buttons to add the fields that you wish to present to the user here. This sort type is useful when your sort components are fixed, but the user needs to switch the order and sequence occasionally.

Description

Enter a description to use for the field name.

Field

Use the ellipsis to select a field name from the *Select Column* dialog.

Order

Select Ascending or Descending from the drop list, to designate an ascending or descending sort order.

Sorting Window

The components of the sorting window generated at run time can be modified here. In addition to the **Title** text and Status Bar **Message,** buttons can also be customized with a variety of options. Click here for additional help.

DynamicNamed

Valid for Report and Process only. This sort type is similar to DynamicSorting above, but is expanded here to allow groups of sort fields to be referenced by a single name. This is powerful with a generic report designed with no default key. Enter different Names, each one with different sort criteria. A single report can now be used for "on the fly" sorting based on the sort groups that you provide.

Name

Enter a group name to use to identify the sort sequence that you design.

Field

Use the ellipsis to select a field name from the *Select Column* dialog.

Order

Select Ascending or Descending from the drop list, to designate an ascending or descending sort order.

Sorting Window

The components of the sorting window generated at run time can be modified here. In addition to the **Title** text and Status Bar **Message,** buttons can also be customized with a variety of options. Click here for additional help.

Add ADO File Extension

The ADO File Extension allows you to include additional ADO file declarations that you have declared within your data dictionary.

Including this extension in any procedure creates a "To Do" in the Data / Tables Pad named "Other ADO File". This interface allows you to include and reference any element of the ADO table without forcing the templates to generate a connection.

Add Other Files too?

Check this box to allow related ADO file definitions to be included with the ADO file that you have added. These files will also be included without forcing the templates to generate a connection

ADO Login Extension Template

The ADO Login Extension template is used with the ADO Login procedure to establish the method and source of connecting to an ADO source.

The following template prompts are provided:

Login Procedure builds the connection string from:

There are essentially two sources of Connection String storage.

1. Select **UserID, Password, and Server prompt at runtime** if the user will be required to login each time the application is loaded (the ADO Login procedure is always the first procedure called by default).

Login details

Number of retries allowed

Enter a number here to set the number of user login attempts.

Display a message if retries exceeded

Check this box to pop up a message box if the user has exceeded the number of login attempts.

Message Box caption

Enter text that will be displayed in the message window’s title bar.

Message Text

Enter the message text to display to the user when login attempts have been exceeded.

Return from procedure after Message call

Check this box to automatically return after closing the message window

Connection string data

After the User is validated, the connection string is generated from one of the following methods:

From Dictionary

Select a table name from the drop list provided. If the table is a valid ADO file type, the connection string will automatically be read and displayed in the text box the follows.

From UDL file

If the user login is successful, the ADO connection string will be read from an external UDL (Universal Data Link) file name that you specify here. Manually enter a valid UDL file, or use the ellipsis to select one from the popup file dialog.

Connection Builder

Select this option, and press the **Connection Builder** button to invoke the data link properties interface and build your ADO connection string. The connection string will be stored by the Login Procedure.

Your entry

Select this option If you wish to manually enter a connection string, or paste it from an alternative source, and a store it locally in the Login Procedure.

2. If you select **UDL File** as the connection string source, a user login is not required. The application attempts to read from the UDL file name you specify.

Check the **Data Link Dialog can be displayed to the user** check box to allow a connection dialog to be displayed when attempting to connect to the data source. If left unchecked, the data connection process is silent (no indication of connection attempt is shown to the user). You should only leave this check box unchecked if you are sure that your ADO connection is from a reliable source.

Call an Error procedure if Connection failed

Check this box if you would like to call a special procedure that will display a custom message and possible reasons for the connection failure. In the **ADO Error Procedure** drop combo, enter the name of the procedure that will be used, or select from an existing procedure in the drop list.

ADO Support Template - Global Extension

Click on a tab above to jump to a sub-topic

What is ADO?

This release of Clarion introduces template support for ADO (Active Data Objects). Your browses and forms can operate with ADO like they would with conventional databases. The big difference is that ADO uses OLE DB/ODBC as the connection means, instead of the native drivers. The ADO templates are designed to provide a similar design interface, as you are already familiar with using conventional database interfaces.

The ADO Global Extension is required for any application that will use ADO. In addition, you can set up Global Connections to ADO that can be used by any ADO-based procedure.

Connections

The ADO Support Global Extension is used to add ADO connections to use throughout your application. Press the appropriate update buttons to add (**Insert**), modify (**Properties**), or remove (**Delete**) an ADO connection.

Oracle Support - Click Here

In the Connections dialog window the following prompts are available:

Connection Object

Enter a name (string) that you wish to use to reference your connection. You can override the default name that is provided by the template.

Connection Class

The Connection Class references the default library used to connect to the ADO service. If you wish to use an alternate third-party class, or a class that you have written, select it from the drop list provided. Otherwise, accept the default class (Cconnection)

Cursor Location

A cursor is a database element that controls record navigation, update ability of data, and the visibility of changes made to the database by other users.

Cursors are set and based on the type of data you anticipate working with, and where the processing will take place.

Don’t set

This tells the ADO connection to use the default cursor method of the particular data source. Unless your site has specific requirements that need to override this, leave this setting as is.

AdUseClient

This value indicates that a client-side cursor supplied by a local cursor library is to be used.

AdUseServer

This value indicates that the data provider or driver-supplied cursor is used.

AdUseClientbatch

For backward compatibility, this value indicates that a client-side cursor supplied by a local cursor library is to be used.

Note that the cursor settings drop list choices are all prefaced with "ad;" these are simply the ADO constant names. The descriptions of each cursor type are provided as "basic" information about the cursor. *Please refer to the documentation provided by your OLE DB Provider vendor and/or DBMS for specific information about your site.*

Connection String – Build From Source

Every ADO application requires a connection string. Select from one of 5 sources to build your connection string from:

From a Login Procedure

Choose this connection source option if you want to create a login procedure for your users to connect to the ADO source. You must use the **ADOLogin procedure** template provided in the ADO template set. The ADOLogon procedure will identify the connection string source (dictionary, UDL file, etc.)

Enter the name of the login procedure to use here.

From Dictionary

Your data dictionary stores ADO connection string information for each file that you imported using the ADO file driver. Press the ellipsis button to choose a **Dictionary Table**. The connection string will be displayed in the text box provided.

You also have the option to save the string read from the Dictionary into a Universal Data Link (.udl) file. Press the **Save Connection String in UDL file** button to write the string contents to an external UDL file.

The first two lines of any universal data link (.udl) file must have exactly this content:

```
[oledb]
; Everything after this line is an OLE DB initstring
```

After these required lines, the remainder of the .udl file consists of a connection string written in the same format used when opening a database from programming code.

UDL file read at runtime

Press the ellipsis button to load an existing UDL file to use as the connection source at runtime.

UDL file picked and read in the template

Select this option if you wish to load an existing UDL file through the current template interface, modify the connection string, and save the modified connection string to a new external UDL file.

Press the ellipsis button to load an existing UDL filename's connection string into the connection builder text window. After your changes are made, you can save the changes to a new UDL file name by pressing the **Save Connection String in UDL file** button.

It is not required to load an existing UDL file. You can create a custom connection string from scratch, or paste it from an alternative source.

Connection Builder

Select this source option to manually build your connection string using the Conncection Builder. There is an option to save this string to an external UDL file.

Override dct Connection String
The Connection dialog window automatically reads the first ADO connection string it finds in the active data dictionary. Check this box to override this default connection string and build another in the **Connection String** text box. Unchecking this box will force a refresh of the default dictionary connection string, and overwrite your changes.

Connection String

Press the **Connection Builder** button to create an ADO connection to a designated data source. The contents (output) of the Connection Builder will be displayed in the Connection String text box. You also have the option of entering a custom (manual) connection string here.

Call Connection Builder

The Connection Builder button calls the Data Link Properties dialog. The specifics of this dialog vary according to which OLE DB provider you choose. You may choose items and fill in four tabs within this dialog.

Note that after calling the Connection Builder, the connection string appears at the bottom of the Connection Properties dialog. The following is a sample of an MS SQL connection for the Northwind sample database using MS SQL Server's OLE DB provider:

```
Provider=SQLOLEDB.1;Password=xxxxxxx;Persist Security Info=True;User
ID1=UserID;Initial Catalog=northwind;Data Source=servername
```

Save Connection String in UDL file

Press the **Save Connection String in UDL file** button to write the current connection string to an external UDL file. You can later use this file as your connection source in multiple applications.

From a STRING variable

Select this option to generate a call to the Connection Object using the variable specified in the **String variable name** template prompt. The developer is responsible to make sure that the variable is initialized before the call to the Connection Object. Use the "*Before calling Connect*" embed point to do this.

Oracle Support

If you are establishing a connection to an Oracle database through the ADO support, check the **Use Oracle Syntax** box. This setting is also used to conditionally omit the setting of the ADO RESYNC command in order to refresh the browse after an update.

To activate special processing for Oracle dates during form updates, additional prompts are provided:

TO_DATE format

Oracle uses a special server function called TO_DATE to process date and time information through the ADO layer. Enter a string constant here that reflects the way that data and time information will be translated through the ADO layer.

Associated CW Date

Enter a valid date picture here, or press the ellipsis to build a picture in the *Picture Editor* dialog.

Associated CW Time

Enter a valid time picture here, or press the ellipsis to build a picture in the *Picture Editor* dialog.

The picture tokens that you select in the CW Date and Time fields MUST MATCH the format string in the TO_DATE format entry. For example, for the following string:

YYYY-MM-DD HH24:MI:SS

Use a @D10- token to match the date portion, and @T4 to mtch the time string. If you change any part of the TO_DATE format, you will need to modify the appropriate CW date or time picture token.

Classes

The Classes tab contains an extensive list of default classes used with the ADO templates. Unless you specifically create a new class definition, or, obtain one from a third-party vendor, you should accept the default class settings.

ADO Process Extension

The ADO Process Extension Template is the core template for all Process and Report procedures that use ADO as a connection layer.

The following prompts are available:

General Tab

Process Prompts

Processed Text
Enter a string of up to 100 characters that will be displayed on the process progress window. If left blank, the default is "Processed"

Timer Value
Enter a numeric value to specify the time slicing interval between the time that rows are being processed, and the update frequency of the progress window. Values are in hundredths of a second. If left blank, the default value is 50 (half second).

Rows to fetch per timer
Enter a number to designate how many rows can be processed in each timer interval. This number should be decreased if the complexity of the process is increased. The default value is 10.

Close Window
Check this box to automatically close the progress window when the ADO process has completed.

Enable Report
Check this box to call a report after the process is completed. This check box will in effect enable the procedure's Report button and the subsequent Report Formatter.

Print Preview
Check this box to call a Print Preview window prior to printing the report.

Detail Filters Prompts
Each Detail band is listed in the Detail Filters list. To restrict printing of a band, highlight it in the list, then press the Properties button. Provide an expression in the Filter field. The band will only print when this expression is true. For example, to force the template to not print a detail, enter FALSE.

Optionally, check the **Exclude unfiltered** box to restrict any other detail band which does not have a filter expression of its own.

Preview Options Prompts
The Preview Options tab lets you control the initial appearance of the report preview window. This tab is only available if you check the **Print Preview** box on the General tab.

Initial Zoom Setting
Sets the initial magnification for the report to one of seven discrete magnification choices. The end user may change the initial setting.

Allow User Variable Zooms?
Check this box to let the end user set custom report magnifications in addition to the preset magnification choices.

Set Initial Window Position
Check this box to enable the four following prompts to set the initial preview window position and size.

X Position The initial horizontal position of the left edge of the window.

Y Position The initial vertical position of the top edge of the window.

Width The initial width of the window.

Height The initial height of the window.

Maximize Preview Window

Check this box to initially maximize the preview window. This supersedes the **Set Initial Window Position**, whose coordinates are applied only when the window is restored to its normal (not maximized) state.

Data

The Data tab control contains prompts that focus on the ADO connection information:

Process Recordset as a parameter

Check this box to disable the ADO connection process (methods) for this procedure. This implies that the information (Recordset) that needs to be processed will be passed to the procedure.

Connection Group:

Connection Object

Choose a Connection object name from the drop list provided. This list should contain the connection object you created in the ADO Global support template.

Use a New Connection

If you do not wish to use any of the Global Connections that are available, check this box to create a new connection specifically for this browse box.

New Connection

Press this button to call the Connection Builder. On the subsequent dialog, enter a new name to use for the new **Connection Object**, and press the **Connection String Settings** button to access the Connection String Settings dialog.

Command Object

The ADO connection layer contains a default command object that is responsible for handling the appropriate data update behavior. If you need to select an alternative object name, enter the value here.

Data Columns/Hot Fields

The Data Columns and Hot Fields tab control provide prompts that allow control of the data elements that will appear in the Process or Report procedure.

The following prompts are presented:

Hot Fields

The Hot Fields list box displays data elements that may not be populated in the report structure, but need to be referenced or updated in the procedure source. Hot fields are normally data elements that are related to the contents of the primary data elements (i.e., Address information, text based descriptions, etc.). Press the **Properties** button to access the following prompts:

Hot Field

Press the ellipsis button to select a field to use as the hot field.

Column is a

Identifies the data element as one that is read directly from the ADO data source (**Table Column**), or one that is a variable defined within the application, like a computed or conditional field (**Expression**).

Use AS

Check this box if the column name is long and verbose, and you need to rename the column to a more descriptive and compacted name. This is useful if you are constructing long or complex SQL statements. This is an option for Table Columns, but required for all Expressions.

Unique Field ID/AS

Enter a unique Field ID to use with the **Use AS** option for Table Columns, or the AS option for an Expression..

Expression

Enter a valid expression to use for the defined Data Column

Expression Data Type
Select a valid data type from the drop list that the ADO layer will translate.

Data Columns

The Data Columns list box displays data elements that have been populated in the Report Formatter. Press the **Properties** button to access the following prompts:

Query Field

Identifies a column as a field that can be queried (searched).

Column is a

Identifies the data element as one that is read directly from the ADO data source (**Table Column**), or one that is a variable defined within the application, like a computed or conditional field (**Expression**).

Use AS

Check this box if the column name is long and verbose, and you need to rename the column to a more descriptive and compacted name. This is useful if you are constructing long or complex SQL statements. This is an option for Table Columns, but required for all Expressions.

Unique Field ID/AS

Enter a unique Field ID to use with the **Use AS** option for Table Columns, or the AS option for an Expression..

Expression

Enter a valid expression to use for the defined Data Column

Default SQL

Because the SQL to access the data is vitally important, the ADO templates allow you to view the generated SELECT statement and customize it if necessary. This also provides a convenient way to customize the process or report ordering, should you wish.

Regenerate SQL

Press this button to reset the original template-constructed SQL statements. This is useful should you need to start from scratch again before customizing your statements. Only enabled when the next prompt is active (checked).

Override SELECT SQL

Check this box to bypass the auto generation of the SELECT statement by the templates. Your custom statement will be substituted in its place. This box also enables the **Regenerate SQL** button, should you wish to reset the statement back to its original value.

Default SQL Select
This text box provides the base SELECT statement. Note that if your process or report contains fields from more than a single table, it will automatically provide for a JOIN.

Important Note: If you've populated a field in the Data Columns tab for which the ADO templates can't resolve the proper syntax and the resulting page doesn't print properly, examine the SQL statement here. Should you find a statement with a blank for the field name (look for an extra comma in the order that the suspect field appears in the data columns list, as in Select fieldname,fieldname,,fieldname...), you may edit the statement here or delete the suspect column from the list.

Unique Key

As stated on the tab control, ADO/SQL requires a unique key to identify a record. Should you have more than one unique key defined in your table, press the ellipsis button to select an alternate key to use.

Default Behavior

The default behavior of the ADO Process control has a much different interface than the standard report or process. You will also notice that the filter capabilities are expanded in the ADO template. The following prompts are available:

Fields Tab:

The Fields tab control provides control of your process or report's filtering and sorting features.

Range or Filter Columns

Press the update buttons to add (Insert), modify (Properties), or remove (Delete) a range or filter column. These are the data elements used to limit the records processes in the Process control. The Range or Filter dialog provides the following prompts:

Column
Select a column name from the drop list provided to use as criteria for the Process/report's filter or range. The columns displayed are those columns you have defined in the Data Columns/Hot Fields dialog.

Use Static Value?
Check this box to use the static value of the column selected. This is the value of the column when the process is first initialized

Negate the Range or Filter(NOT)

Check this box to apply the NOT clause to the selected column. This will allow records to be selected only if the column value evaluates to a blank or zero.

Range Limit Type

Specifies the type of range limit to apply. Choose one of the following from the drop-down list.

Single Value

Lets you limit the filter criteria to a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

Range of Values

Lets you specify upper and lower limits. Specify the variables containing the limits in the **Low Limit Value** and **High Limit Value** boxes.

Less Than

Lets you limit the records read to all records less than a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

Great Than

Lets you limit the records read to all records greater than a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

IN

Lets you limit the records read to all records that match the contents of a single value.

Begins with

Lets you limit the records read to all records that begin with the contents of a single value.

Ends with

Lets you limit the records read to all records that end with the contents of a single value.

Contains

Lets you limit the records read to all records that begin with the contents of a single value.

Sort Columns

Press the update buttons to add (Insert), modify (Properties), or remove (Delete) the process controls default sort elements. These are the data elements used to alternately sort the records processed by the control. The SQL tab control generates the ORDER BY clause.

Add Primary Key Fields
Press this button to automatically include all fields defined in the primary key as sort columns.

Add a Key Field
Press this button to select any key from the primary table, and apply it to the sort columns.

The **Sort Columns** dialog also provides the following prompts:

Column
Select a column name from the drop list provided to use as criteria for the browse filter or range. The columns displayed are those columns you have defined in the Data Columns/Hot Fields dialog.

Direction
Choose ASC from the drop list to designate an ascending sort, or DESC to specify a descending sort.

SQL Tab Prompts

The SQL tab control displays SQL statements that are generated, based on the settings displayed on the Fields tab.

Essentially, the settings of the Range or Filter are used to generate the SQL WHERE clause, and the Sort Columns setting are used to generate the ORDER BY clause.

Regenerate WHERE SQL

Press this button to regenerate the template SQL WHERE statement that is constructed based on your default settings.

Override WHERE SQL

Check this box to override the default template generated WHERE clause.

Enter a WHERE clause to filter this list

The text in this box provides the Where clause, which is concatenated to the Select statement.

Regenerate ORDER BY SQL

Press this button to regenerate the template SQL ORDER BY statement that is constructed based on your default settings.

Override ORDER BY SQL

Check this box to override the default template generated ORDER BY clause.

Enter the ORDER BY clause

The text in this box provides the Order clause, which is concatenated to the other parts of the statement.

Test SQL Query

Press this button to test the SQL statement. If a statement clause is not overridden, the statement will be regenerated based on the template's current settings.

Conditional Behavior

The Conditional Behavior tab duplicates the prompts and resultant functions found on the Default Behavior tab, with the addition of a **Condition** prompt:

Condition

Enter any valid Clarion expression.

Classes

Use the Classes tab to override the global settings for the Class. See Classes Tab.

BLOBInControl Extension Template

The BLOBInControlExt extension template allows you to display and/or edit a BLOB from a selected table.

Normally, BLOB processing involves populating a local or global variable that will load the BLOB contents from a selected table. As the BLOB is updated, the contents of the control is transferred back to the BLOB and stored in the table. This template is designed to handle all of the processing that is necessary to do this.

The use of this template is valid in Browse, Form, or Report procedures to retrieve (read) a BLOB's contents and assign it to a control. In addition, you can also save (write) the BLOB's contents in the Form procedure.

A list box is provided to allow you to enter multiple BLOB controls. On the *BLOB Fields Properties* window, the following template prompts are provided:

BLOB Field
Select from the drop list the target BLOB field defined in the table that will be displayed and potentially updated (if used on a Form).

Control
Select the Field Equate Label, from the drop list, of the control that will be used to display and/or update the BLOB contents. This control must be either a text or image control, depending on whether the type of information contained in the BLOB column is text or Binary (i.e. Image) data.

If this is a text control, the use variable for the control must be a string type variable and it must be large enough to hold the largest potential Blob data in the associated Table that you plan to update.

When Image does not exist use the following image – Image File:
If the control is an image control, enter the name of a valid image file (or press the ellipsis button to select one) that will display if an image does not exist in the target BLOB control.

Resize to control size
Check this box if you would like the BLOB's contents to resize to fit the image control. If not checked, the default size as stored in the BLOB is used. Using the default could cause the image to display cropped if the image control is too small.

Condition to assign from control to BLOB
Enter an expression in the **Condition** field which will force the BLOB to be updated from the target control *only* when the expression evaluates to TRUE. Press the ellipsis button to select a variable to use in the expression.

By default, this setting is TRUE, which means that the BLOB field will *always* be updated when the Form is Accepted, even if the BLOB contents did not change. To improve program performance and network traffic, enter a condition that will cause the BLOB to be updated only when necessary.

For example:

```
?Text{PROP:Touched}=True !was the text field modified?
```

or

```
CLIP(LOC:ImageFileName)<>'' !was an image loaded into BLOB?
```

CleanCloseDown Global Extension

(ABC Template Chain Only)

This extension template is used to include the CleanCloseDown( ) procedure prototype that can be used in any procedure to cleanly close down your application.

A clean closedown is defined as the proper and automatic shutdown of any application, regardless of the state of the application. For example, if a user leaves the office with an application running, and a form procedure is open, the **CleanCloseDown** procedure, when called on a timer event, can shut down the application *without user intervention*.

This global extension automatically includes a local extension for every procedure in your application that incorporates a WINDOW structure.

To use this template, simply call the **CleanCloseDown()** procedure from any embed point. Normally this would be from a TIMER event on any window.

The **CleanCloseDown**( ) procedure is simple, and takes advantage of the proper method to close down a program in a multi-threaded environment with the proper use of the NOTIFY language statement:

```
CleanCloseDown PROCEDURE()
CODE
MESSAGE('BYE')
GLO:CleanCloseDown = True
NOTIFY(NOTIFY:CloseDown,GLO:CleanCloseDownMainThread)
```

When using this template in your application, the following global variables are internally defined:

```
GLO:CleanCloseDown            BYTE(0)
GLO:CleanCloseDownMainThread LONG
```

The only template prompt available is the ability to **Disable Clean Close Down** when testing its compatibility with other third-party templates.

In multi-DLL applications, you will also need to include the global extension in any DLL target application that will need to use the **CleanCloseDown** procedure.

Date Time Display extension template

This extension template adds to the functionality of a procedure template, allowing you to display the time and/or date in the status bar, or a control.

The prompts for this template are accessible through the **Procedure Properties** dialog of a template that includes this extension. A **Date and Time Display** button appears in the dialog of the procedure template.

The options that appear in the Date and Time Display dialog are divided into two group boxes, **Date Display** and **Time Display**:

Display in Window
Check the box or boxes to add the date or time display to your window.

Date/Time Picture
Choose a date and/or time display picture from the drop down list. The list displays examples, such as "October 31, 1959," and "5:30P.M."

Other Date/Time Picture
Type in a picture of your choice, if the picture type you wish does not appear in the list. **See Also:** Date Picture Tokens.or Time Picture Tokens

Show the day of the week before the date
(Date only) Optionally displays the day of week.

Location of Date/Time display
Choose between displaying the date and/or time on the status bar, or in a control.

Status Bar Section
When specifying the Date or Time should appear on the status bar, specify the status bar section.

Date/Time Display Control
When specifying the Date or Time should appear in a control, choose the control from a drop down list of field equate labels for the window.

DbAuditing

The DbAuditing template is a global extension used to track updates to specific data tables. This extension is added to the global properties of an application.

The DbAuditing template provides the following prompts:

Log Files

This tab contains a list box that lets you define log files and options for the files used in an application. Add a new log file definition by pressing the **Insert** button. This displays a dialog where you name the log file and define its contents. Use the  and  to change the order of the log files.

File to Log
Choose a file from the drop list that consists of all files defined in the dictionary. Any updates to this data file are written to its associated log file.

LogFile Name
Specifies the name of the log file to update.

Username Logging

The username logging option is used to track which user is making the update to the specific data file. It does this by adding an extra column to the log file for the user's login name.

Username variable
Specifies the variable which contains the current user. The value of this variable gets written to the log file to indicate the user who made the change to the data file.

Username Header
Specifies the header to be used in the log file for the user column.

Username picture
Specifies the format to be used for the user column in the log file.

File Fields to Log

This option gives the ability to customize a log file for each data file. For example, you may wish to log changes to certain fields and allow changes to others without logging them. Press insert to select a field to write to the log file. Use the  and  to change the order that the fields are logged to the log file.

Field
Specifies the field name for the data that is writen to the log file.

Field Header
Specifies the header to be used in the log file for the data column.

Field Picture
Specifies the format to be used in the log file for the data column.

Miscellaneous Fields to Log

This option gives the ability to customize a log file for miscellaneous data. Press insert to add a field to the log file. Use the  and  to change the order that the miscellaneous fields are logged to the log file.

Field
Specifies the field name for the data that is writen to the log file.

Field Header
Specifies the header to be used in the log file for the data column.

Field Picture
Specifies the format to be used in the log file for the data column.

ExtendProgressWindow

The ExtendProgressWindow template adds functionality to Process and Report procedures. It is designed to do two things:

- Give you precise control over the visual feedback you provide end users for (small) Process and Report procedures.
- Allow Process and Report procedures to operate in two separate modes--all records mode and single record mode (current value range-limit).

You can use the ExtendProgressWindow template to delay or to completely suppress the progress window for a Process or Report procedure, and you can optionally specify a wait cursor. In single record mode, you can suppress the progress window, the print preview, or both.

The ExtendProgressWindow template provides the following options.

Delay Showing Window
Enter the number of seconds to hide the progress window. For example, you may want to hide the progress window for 3 seconds so that processes or reports that finish within 3 seconds limit never show a progress window.

Wait cursor
Check this box to display a wait cursor (hour glass cursor) for the duration of the process or report. For small/short processes and reports, your end users may prefer a simple wait cursor over a progress window. On completion, the procedure restores the cursor to its previous state.

Single Shot
These options are available only for Processes and Reports that specify a key in the **Data / Tables** Pad.

Single record
Check this box to allow the Report or Process to operate in its normal mode (process all records), or to operate in single record mode (current value range-limit) when GlobalRequest is set to ProcessRecord (see *Procedure Templates--Inter-Procedure Communication* for more information on GlobalRequest).

If your Report or Process procedure uses a non-unique key, you can process all records with the current key value!

The BrowsePrintButton template primes the range-limit field and calls procedures in this single record mode (see *Control Templates--BrowsePrintButton*).

Use Progress
Check this box to display the progress window in single record mode. Clear the box to suppress the progress window in single record mode.

Use Preview
Check this box to provide the print preview in single record mode. Clear the box to suppress the print preview in single record mode.

FormVcrControls extension template

This Extension template adds functionality to a Form procedure by enabling navigation and field history with the *FrameBrowseControl* VCR buttons. See *FrameBrowseControl* for more information on these buttons and their operation.

Essentially, the *FormVCRControls* Extension provides a "scrolling" Form. You can display, add, delete, or edit many records without returning to the calling Browse to select a new record. However, the keys and filters implemented in the calling Browse procedure do control the navigation of the Form. For example, you can only navigate to records that meet the Browse range limit and filter conditions, and when you navigate to the "next" or "previous" record, the Browse key determines the sequence in which the records appear.

For Form procedures generated by the Application Wizard, if the Form procedure also contains a *BrowseBox*, the *FrameBrowseControl* buttons control the Form when the "form" tab is selected, and they control the *BrowseBox* when the "browsebox" tab is selected. See also *SetToolbarTarget* code template.

FileUserTags

This global template extension identifies certain tables and columns that possess a particular **User Options** tag defined in the dictionary. The template then marks or assigns that tag, and allows the programmer to reference the tag *in place of* the physical table or column name. This allows the programmer to write code using the **User Options** tags in the dictionary in place of the actual names used to define them.

The FileUserTags template provides the following prompts:

Interesting File Tags

Specifies defined table tags from the dictionary to use as a reference to the table that contains the tag.

Interesting Field Tags

Specifies defined column tags from the dictionary to use as a reference to the columns that contain the tag.

FrameExtension Template

The FrameExtension extension template adds two powerful features to any Application Frame; the ability to shut down any application when Windows shuts down , and the ability to dock your Clarion application on to the system tray.

The following options are available:

Enable ShutDown
Check this box to allow your application to close down when the Windows operating system is also shut down.

Enable Tray Icon
Check this box to allow your application to be minimized in the system tray instead of the system toolbar. By default the icon used in the system tray is the same icon that is assigned to the application frame.

SetToTrayOnLoseFocus
This option allows an application to be minimized to the system tray when losing focus.

This option is only active if no MDI child windows are opened. This "lose focus" option is designed to be used in applications with only one window (like the IPDRV service manager).

Tray Icon Tooltip:
Enter a valid string value to use as a tool tip. When your mouse moves over the system tray icon, the tool tip is displayed.

Press the **Tray Icon Mouse Right Button Menu** button to access an additional dialog that allows you to add a context menu to the system tray icon. Right-clicking on the system tray icon activates the popup menu.

The Context Menu interface allows you to construct custom menu items to add to the popup menu used with the system tray icon. You can select from a list of actions to attach to each menu item that you create.

Press the appropriate button to add (**Insert**), modify (**Properties**), or remove (**Delete**) a context menu item. The following options are available or the subsequent dialog.

Is a separator?
Check this box if you would like to add a separator line to your context menu. There are no other prompts associated with this action.

Menu Text
Enter the name of the context menu item to display.

When Pressed

No Special Action
The menu item will have no template-generated action associated with it. This option normally is used when you are using an embed point to control this menu's action.

Execute Routine
Enter a valid **Routine Name** here. Of course, you must have the ROUTINE defined within the scope of this procedure.

Post Event to Control
Select a valid Field Equate from the drop list to post an event to the **Control.** Select an *Accepted* or *Selected* **Event** to post to the control from the drop list provided.

Call a Procedure
The Context Menu interface uses the standard interface for calling a procedure. If you need more detailed help here, see the following topic.

Run a Program
The Context Menu interface also uses the standard interface for running a program. If you need more detailed help here, see the following topic.

Embed and Class Support

The FrameExtension template is internally supported by the **WindowExtenderClass** ABC compliant CLASS, found in WINEXT.INC.

Although there is currently no detailed documentation available for all properties and methods in this class, the following information is provided as a quick primer in using some of the more important properties and methods.

The FrameExtension template derives a FrameExtension CLASS, which is initialized in the ThisWindow.Init WindowManager method.

After this method is initialized, you can use the **SetAllowShutDown** and **SetAllowTrayIcon** methods to control the support for Shutdown and the TrayIcon respectively. For example:

SetAllowShutDown(1) !Allow Windows shutdown while application is still active

Both methods use a BYTE value to set or disable the appropriate function.

The FrameExtension template also generates the following virtual methods for your use, through appropriate embed points:

TrayIconMouseLeft( )
Process the single click mouse event on the tray

TrayIconMouseRight( )
Process the single right-click mouse event on the tray

TrayIconMouseLeft2( )
Process the double click mouse event on the tray

TrayIconMouseRight2( )
!Process the double right-click mouse event on the tray

ProcessShutDown( )
Process the Window Shutdown while the application is active. If the return value of this method is TRUE (1), the shutdown will continue. If the return value of this method is FALSE (0) the shutdown will not continue (it will be stopped).

Finally, there are two other methods that you can use at any time after the class has been initialized:

RestoreFromTray( )
If the application is present on the system icon tray, a call to this method will remove the icon, unhide and show the active window.

SendToTray( )
If this method returns TRUE (1), the process of adding the icon to the tray and hiding the window was successful.

Global Cooperative Threading Extension

The options for this extension are made up of a single check box.

Enable Global Cooperative Threading

Check this box to switch your application to use a cooperative threading model in place of the preemptive model.

More information on these models (including advantages and disadvantages) can be found in the Preemptive and Cooperative Thread Models topic.

This template automatically includes a Preemptive Procedure Extension in all procedures defined in your application. This allows you exact and convenient control over each procedure's thread models.

For example, if the majority of procedures will need to use the cooperative thread model, check the global option, and use the Preemptive Procedure Extension to enable the few exceptions. On the contrary, you can turn off the Global Cooperative Threading model, which makes all procedures preemptive by default, and enable Cooperative Threading by turning off the individual procedure's Preemptive Procedure Extension.

Classes

The Cooperative Threading model uses the default ThreadLocker and CooperationClass classes. If you wish to include a different set of classes to use in place of these defaults, enter the new name in the prompts provided, or press the appropriate ThreadLocker and CooperationClass buttons to override the default classes behavior.

MDI Synchronization - Global Extension

This Global template has a single check box, which is used to include the MDI Synchronization Class libraries. By default, all procedures will include an MDI Synchronization Object, with the exception of Report procedures and Clarion Chain applications not using the ABC Class Libraries.

You can toggle this check box to test the application with or without the Synchronization objects.

Who needs to use this template

Based on our analysis, and on communications from Microsoft, two significant factors were identified that correlate with problems experienced in multi-threaded MDI applications:

- The presence of Menus on the MDI child window.
- Very fast opening or closing of MDI windows.

The MENU on child windows is generally only a factor when the MDI windows are being opened Maximized. These are some "worst-case" scenarios:

MDI child windows that are opened in a LOOP and are either maximized initially, or the program maximizes them directly on the basis of information stored in a user configuration files such as an INI file or in the Windows registry.

The program does something that requires the OS to update internal MDI structures, and before that can complete initiates another operation that also requires the same structure to be updated. For example, opening other windows, on the event EVENT:Sized. At the point when this event is processed, the Operating System may not have completed update of internal structures associated with the first MDI window, thus the internal OS structure becomes corrupted.

The program uses Menu’s on the MDI Child windows and the windows are opened Maximized in a loop.

Effects you can see

The effects that you may encounter from code like the above are:

- The Menu bar on the FRAME is updated incorrectly
- The Windows list frequently contains fewer windows than are opened
- Under W9x/ME: frequent GPFs either during maximization or if the end-userchooses to tile or cascade opened windows.
- Ghost caption buttons on the FRAME which can cause a lockup on opening-maximization of numerous child windows

Sample "bad" code

Now let’s look at some sample problem code:

```
MyCounter = GETINI( 'MDIsettings', 'NumberOfWindowsToOpen')
LOOP MyCounter times
START(MDIwindowProc, 25000)
END
```

This code is very likely to have problems, particularly on slower machines, and even more so on less robust OS's like W98/ME. It is generally a bad programming practice to write code like the above, especially if your application must run on W9x/ME OS's. However, the results from our tests using the 6.1 RTL and the new "MDIsynchro" class/template this code will generally succeed, --- but it is not guaranteed because so much can depend on factors outside the control of your program; things such as Operating System, CPU speed, available memory/resources, number of other running processes may all affect your program.

Summary

Most of the problems found can be avoided with the use of the 6.1 Runtime Libraries and the new MDI synchronization template, however some problems may also require changes in your coding style.

MDI Synchronization - Procedure Extension

This template has a single check box, which is used to disable MDI Synchronization for this prodecure. By default, all procedures will include an MDI Synchronization Object, with the exception of Report procedures and Clarion Chain applications not using the ABC Class Libraries.

You can toggle this check box to test the procedure behavior with or without the Synchronization object applied.

See Also: MDI Synchronization - Global Template

Preemptive Procedure Extension

The Preemptive Procedure Extension is included in each of your application's procedures if you have included the Global Cooperative Threading Extension template in your application.

The sole purpose of this template is to provide an override to the Global settings. If the Cooperative threading model is active, check this box to allow this procedure to use the Preemptive model instead. On the contrary, if the cooperative threading model is turned off on the global level, turn off the check box here to allow this procedure to use the cooperative threading model.

More information on these models can be found in the Preemptive and Cooperative Thread Models help topic.

Process Transaction Frame extension template

This template is available for any Process procedure in the ABC template family. It is used to override the application's default transaction frame support for any RI operations referenced by the primary table of the process procedure.

The following template prompts are provided:

Generate one transaction for each record read

By default, the Process Transaction Frame extension will enclose a transaction frame around the entire process (Starting the transaction process in the OpenReport method, and finishing the transaction process in the CloseReport method). This gives you an "all or none" default option.

Check this box to allow a transaction frame to be applied to *each* record read. This allows you to create partial updates in the process procedure, in the event that any errors occur on other records processed.

You can even get a greater control on how many records get "framed" with the inclusion of the Process Transaction Frame Checkpoint code template in the embed point of the TakeRecord method. This checkbox must be on for the frame checkpoint to be generated.

Include Children in the transaction?

The primary table named by the process procedure will *always* be included in a transaction frame on any update operation. This option allows you to specify if you want to include any children that are related to the primary table (via the dictionary). Select *Always* if you want to include all child tables in all update operations. Select *Never* if you want to exclude all related child tables from all database operations on the primary table.

List of tables included in the transaction

This list box allows you to include other tables that are not default child tables specifically related to the process procedure's primary table. An example of this may be a "history" table that periodically gets updated under user control. Pressing the **Insert** or **Properties** button displays the following additional prompts:

Table

Enter the label name of a table that you wish to include in the transaction frame, or press the ellipsis to select a table from the subsequent *Select Table* dialog.

Include Children in the transaction?

The options shown here are described in detail in the similar prompt documented above.

Classes Tab

Provides the standard template class interface. By default the supporting TransactionManager object generated by the template is named *Transaction*.

Record Validation extension template

This Extension template adds functionality to a Procedure template by enforcing data dictionary-defined control value validation. It also lets you specify controls to exclude from validation.

The prompts for this template are accessible through the **Procedure Properties** dialog of a template which includes this extension. A **Record Validation** group box appears in the dialog of the procedure template.

Validate when the control is Accepted
Specifies that validity checking occurs when the control generates an EVENT:Accepted, which occurs when the end user completes or moves the focus from the field.

Validate during NonStop Select
Specifies that validity checking occurs when any control value changes if the window is in AcceptAll (Non-Stop) mode and has focus.

Color Fields rather than Selection
Check this box to change the column or prompt color when an invalid entry occurs.

Field Color when Invalid
Press the ellipsis button (…) to select a color to apply to the column when an invalid entry occurs.

Prompt Color when Invalid
Press the ellipsis button (…) to select a color to apply to prompt when an invalid entry occurs.

Show Message when fields are Invalid
Check this box to display a text message when an entry is invalid.

Message to Display
Specify the text for the message to display when an entry is invalid.

Control to place Message in
Select from the drop down list, the control that will display the invalid entry message.

Do Not Validate
Opens the Do Not Validate dialog, which lets you select fields from a drop down list. The fields you choose will be excluded from validity checks.

ReportChildFiles

The ReportChildFiles template adds functionality to Process and Report procedures. This extension template provides a simpler, more efficient, more controllable alternative to setting a chain of related files in the Data / Tables Pad and having the Report or Process template produce a single multi-tiered VIEW.

The ReportChildFiles template lets you name only the primary file and any lookup files in your procedure's **Data / Tables Pad**. The template generates code to read (and optionally print a separate DETAIL for) the related child-file records for each primary file record. We recommend the ReportChildFiles template for the typical invoice headers/invoice lines scenario.

Multi-tiered View

Suppose you have an invoice header file and an invoice detail file. You want to print out a header and then a line for each detail. This is somewhat tricky to do with a single view and there are some limitations and inefficiencies with this approach. You must populate each header (parent) file field into a group HEADER and each detail (child) field into a DETAIL. The limitation is there are no events and no embed points to use when the parent record prints (because it is simply a group break). The inefficiency is that additional GETs are done on parent file lookups for every child record even though the parent record is unchanged. Plus, for SQL you must use a left outer join (inefficient) to force parent headers to print when there are no associated detail lines.

ReportChildFiles

With the ReportChildFiles template you can simply populate the header (parent) as the primary file with its own DETAIL, then populate a second DETAIL for the detail (child) file. The primary view is then read record-by-record (lookups done only once for each parent record) and the child view is range-limited on the parent file linking fields. The Process Manager Method TakeRecord embed point provides an access point for *both* parent and child records. ProcessClass.TakeRecord is called for each record (parent or child), and ProcessClass.ChildRead indicates which file/record is active. See *ProcessClass* for more information.

Using the ReportChildFiles Template

The ReportChildFiles template provides the following options.

Parent File

Type the label of the parent file, or press the ellipsis button (...) to select the parent file from the **Select Table** dialog.

Detail

For Report procedures, select the USE attribute (field equate label) of the REPORT DETAIL structure to print for each child record.

The Detail drop-down list shows DETAIL structures with USE attributes, so populate the DETAIL first, and add a USE attribute.

Data / Tables Pad <To Do>

Insert the *child* file to process for each parent file record.

Classes Tab

Use the Classes tab to override the global ViewManager setting. See *Template Overview--Classes Tab Options--Global* and *Local*.

RunCommandLineProc

The RunCommandLineProc global template extension provides a simple way to check for command line parameters when the program is run. Several command line parameters may be defined and sorted in the order they should be checked at runtime.

The RunCommandLineProc template provides the following prompts:

Log All Errors Silently
Check this box to disable the ErrorManager error windows while running the procedure nominated on the command line. Errors are written to a log (.TXT) file. This avoids the need for any user intervention.

Procedure Name Flag
Specifies a special command line parameter to check for when the program is started. A variable can be specified here if it is preceded by an exclamation point (!).

Call generated procedure
Check this box if the presence of the command line parameter on the command line will cause a procedure generated from the application to be called.

Procedure To Run
Specifies the procedure to be called when the command line parameter is passed to the program on the command line. Choose a procedure from the drop down list if a generated procedure will be called. Enter the name of the procedure in the entry column if it is am external procedure to the application.

Save Button Transaction Frame extension template

This template is available for any procedure that uses the ABC Save Button control template. It is used to override the application's default transaction frame support for any RI operations referenced in the target procedure.

The following template prompts are provided:

Include Children in the transaction?

The primary table named by the Save Button control template will *always* be included in a transaction frame on any update operation. This option allows you to specify if you want to include any children related to the primary table. Select *Always* if you want to include all child tables in all update operations. Select *Only on Delete and Change* to include the child tables in the transaction frame only during those specific update operations. Select *Never* if you want to exclude all related child tables from all database operations on the primary table.

List of tables included in the transaction

This list box allows you to include other tables that are not default child tables specifically related to the Save Button control template's primary table. An example of this may be a "history" table that periodically gets updated under user control. Pressing the **Insert** or **Properties** button displays the following additional prompts:

Table

Enter the label name of a table that you wish to include in the transaction frame, or press the ellipsis to select a table from the subsequent *Select Table* dialog.

Include Children in the transaction?

The options shown here are described in detail in the prompt documented above.

Classes Tab

Provides the standard template class interface. By default the supporting TransactionManager object generated by the template is named *Transaction*.

Set DLL Image Base Memory Address

This template extension allows you to rebase your application's DLLs. Rebasing is a technique used to modify the base memory address of your DLL.

The base address of a DLL is the preferred location in your application's virtual memory address space where the loader attempts to place the DLL. It is generally specified at link time and used by the linker to write address pointers into the DLL binary . If a DLL cannot load at its preferred base address because the memory region required is already occupied, the loader relocates the DLL elsewhere in virtual memory, then updates all of the address pointers in the DLL to adjust them for the new base address. Making these changes can be time consuming, because the image must be copied to the page file and modified for the new address. This also consumes page file memory until the application is closed and results in the DLL not being shared. A properly based DLL can be demand loaded from disk, so it consumes no page file memory and can be shared. If you have a large application that uses many DLLs, it is important for each DLL to have a different base address to minimize load time and memory use.

It is best to base DLLs from the top of the address range down, instead of from the bottom up. Dynamic memory is allocated from the bottom up and if a DLL tries to load where dynamic memory has been allocated, it will be relocated, just as if a DLL was loaded at that address.

This template is only valid for applications that use DLL as their target output type. EXE and LIB target types are not affected by this template.

Choose from the following prompts:

Select method
Choose *Specify manually*, if you would like to manually assign the DLL's new Image Base Address, or *Choose from list* to provide a valid set of choices to choose from.

Manual Address

This tab contriol allows you to specify a manual image base memory address for your target DLL.

DLL Base Address in hex
Enter an eight digit memory base address in hexadecimal format.

The DLL Base Address must be between 02000000 and 6FFF0000 and end with 0000.

Your EXE and the Clarion libraries use addresses 00400000 to 02000000, so this template will prevent you from using any address below 02000000

Use a Process viewer (like Dependency Walker) to verify that your address space as no conflicts. If you do cause a conflict, the Windows loader will resolve it for you automatically.

IMAGE_BASE ########h will be added to the Export file with the address specified if the target file is a DLL.

Choose Address

This tab control allows you to select a memory address from a list of image base memory addresses to use for your target DLL.

DLL Base Address in hex
Enter an eight digit memory base address in hexadecimal format.

Select a different base address for each DLL you use in your application.

The listed addresses are on 1 MB boundaries and should allow room for large DLLs. If your DLLs are over 1 MB in size, then skip an address to allow more room.

Use a Process viewer (like Dependency Walker) to verify that your address space as no conflicts. If you do cause a conflict, the Windows loader will resolve it for you automatically.

IMAGE_BASE ########h will be added to the Export file with the address specified, if the target file is a DLL.

Version Resource extension template

The Version Resource extension template gives you the ability to create and implement a Version Information Resource file to use with your application target (executable or DLL).

The information generated by this Version Resource file can be found by right-clicking on your executable or DLL and selecting the Version tab.

Version Information is controlled by the Project System. This template automatically generates a version file using the *applicationname.version* format, and includes it in the project. You can learn more about this process by clicking on the Version Information Resource Files topic.

The following template prompts are provided:

General

Company Name
Enter the name of the company that produced the file (e.g., - *SoftVelocity Corporation*)

Legal Copyright
Optionally enter any copyright notices that apply to the file. Include any pertinent text, symbols, and dates. (e.g., - *Copyright © SoftVelocity 1994-2003*)

Legal Trademarks
Optionally enter any specific trademark or registered trademarks that apply to the file.

Comments
Optionally enter any information that may be used for diagnostic or general information purposes.

There is a setting in the Version Resource File that is not directly controlled by the templates. This is the "OriginalFilename" VALUE that is set from the value of the Project "Target file".

For example, if your target file is set to "CWHH60.DLL", then the .version file will contain the following line:

VALUE "OriginalFilename", "CWHH60.DLL\0"

In the Clarion LIBSRC folder, there is a *default.version* file the contains default information for the following template prompts:

VALUE "CompanyName", "Default Company\0"
VALUE "LegalTrademarks", "Trademark\0"
VALUE "LegalCopyright", "Copyright (c) 2003\0"
VALUE "Language", "U.S. English\0"
VALUE "CodePage", "Multilingual\0"

These values can be removed or edited as needed. Do not delete the *default.version* file.

Product Version

Use External Product Information
Check this box if you would like to include the product information from an external source file. Enter a **Source Version File** to include. This file must conform to the proper Version string information standards required by the resource file.

This is useful when compiling multiple applications that are all used with a product theme, like a payroll or accounting package.

Product Name (Required)
Enter the name of the product that is associated with this file. For example, you may distribute a special utility or tool that is associated with an overall accounting software package.

Product Major Version (Required)
Enter an integer that represents the major version of the product that this file is distributed for.

Note:

Product and File Version information use the following format:

major.minor.sub.buildnumber
For example, if major = 6, minor = 0, sub = 1, and build number = 555, the product version results as 6.0.1.555

Product Minor Version
Optionally append a letter or number to the minor version.

Product Sub version
Optionally append an integer to the sub version.

Use Generated Build Number
Check this box to allow the template to generate a build number for you. The template maintains a text file with a .BLD extension that increments the build with each source generation. You can manually edit this file if you wish, or only check this box when you are ready to update versions.

Product Build Number
Enter a valid integer here that represents the build number.

File Version

Internal Name (Required)
Enter the internal name of the file. This could be a module name if the file is a dynamic linked library. In most cases, this name should be the same name as the executable.

File Description (Required)
Enter the name of the file description that the users will see. This name appears directly on the Version tab.

File Major Version (Required)
Enter an integer that represents the major version number of this file. This name appears directly on the Version tab.

Note:

File Version information uses the following format:

major.minor.sub.buildnumber
For example, if major = 6, minor = 0, sub = 1, and build number = 555, the product version results as 6.0.1.555

File Minor Version
Optionally append a letter or number to the minor version.

File Sub version
Optionally append an integer to the sub version.

Use Generated Build Number
Check this box to allow the template to generate a build number for you. The template maintains a text file with a .BLD extension that increments the build with each source generation. You can manually edit this file if you wish, or only check this box when you are ready to update versions.

File Build Number
Enter a valid integer here that represents the build number.

Pre-Release Build
Check this box to mark this file as one that is part of a pre-release build. Although your user will not see anything special generated by this option, it is still useful for developers using the internal Version Info functions during installs and updates. This option primes the internal VS_FF_PRERELEASE flag.

Locale

Language

Select a user's target language in the drop list provided. The language you select will appear in the Version information tab

Code Page
Select a valid Code Page to assign to the Version Information. The default value is *multilingual*.

Clarion

Include Clarion Version Information
Check this box if you would like to include Clarion specific information into the generated Version output. You can override the **Clarion Version, Template family,** and **Template version** information that is extracted by the template.

User
Additional string information that will appear in the Version Info list located on the Version tab can be added here (up to 4 entries). Enter a **label** and corresponding **value** in the template prompts provided.

WindowResize extension template

This Extension template lets the end user resize windows that have traditionally been fixed in size due to the controls they contain (List boxes, entry controls, buttons, etc.).

The template generates code to reposition the controls, resize the controls, or both, when the end user resizes the window.

Tip

To allow window resizing you must set the WINDOW's frame type to Resizable and you must check the Immediate box to add the IMM attribute to the WINDOW.

Overriding Resize for a specific control

Resizer Configuration Options

Resize Strategy

Specifies the method for resizing and repositioning the controls to fit within the new window size.

Centered

With this strategy, all controls will keep its size and the position will change to stay centered in the window.

Use Anchor:

This option by default works the same way as the **Centered** option, but also moves additional anchor strategies to each control's **Action** Tab. The resizing strategy works using "Anchors" to define how the control will behave on the resize with regard to its parent control.

Parent controls can be a TAB, OPTION, GROUP, or a WINDOW. For example, a RADIO control will anchor based on the OPTION resize direction.

Note:

As far as all resize strategies are concerned, a control's parent control is identified by its *position* on the window and not how it is defined *structurally*. For example, if a BUTTON is defined outside of a TAB control, but its default physical position is located *within* the TAB control, the TAB control is the actual parent control for that BUTTON.

Anchoring is based on what window edge is resized. For example, if a control is set to *Anchor:Left*, resizing the window's left edge will move the control proportionally to the left. If a control is set to *Anchor:Right*, resizing the window's right edge will move the control proportionally to the right.

The anchors define which edge of the parent control the control will be bound. Possible values for Anchor Strategy are *Anchor:Right*, *Anchor:Left*, *Anchor:Top*, and *Anchor:Bottom*

To make these options easy to use, setting this strategy enables the **Set Resize Control's Anchors** button located on each control's **Actions** tab.

This dialog allows you to specify which edge of the parent control that the target control will be bound (anchored) to.

In addition to the anchor buttons shown above (T,B,L,R), there are additional buttons (^,V, <, >) that allow the control's target edge to resize proportionally in the active vertical or horizontal direction. For example, if a window is resized on top, what control edge will "grow" proportionally based on the amount of sizing?

To better illustrate the use of anchor and the grow buttons, click here for a detailed example.

Resize

The generated code scales all window coordinates by the same amount, thus preserving the relative sizes and positions of all controls. That is, all controls, including buttons and entry fields get taller and wider as the window gets taller and wider. Window fonts are unchanged.

Spread

The generated code applies the following strategies to the respective control types:

Button
Only the horizontal position (X coordinate) is scaled with the window; width, height, and vertical position are unchanged.

Radio Button
Horizontal and vertical position are scaled with the window, but width and height are unchanged.

Check Box
Horizontal and vertical position are scaled with the window, but width and height are unchanged.

Entry
Width, horizontal and vertical position are scaled with the window, but height is unchanged.

Combo Box
Width, horizontal and vertical position are scaled with the window, but height is unchanged.

Spin Box
Width, horizontal and vertical position are scaled with the window, but height is unchanged.

Drop Combo
Width, horizontal and vertical position are scaled with the window, but height is unchanged.

Other Controls
All coordinates are scaled with the window.

Surface
Makes the most of the available pixels by positioning other controls to maximize the size of LIST, SHEET, PANEL, and IMAGE controls. We recommend this strategy for Wizard generated windows.

Even though list boxes may be resized, the column widths within the list box are not resized. However, the right-most column does expand or contract depending on the available space.

Don't Alter Controls
Controls are not resized when the window is resized.

For this strategy, you may add the SCROLL attribute to each control plus the HVSCROLL attribute to the window to provide a 'moving window' over a larger page.

Restrict Minimum Window Size
Check this box to specify a minimum window height and width. This lets you enforce a minimum reasonable size of the window based on the size and number of controls on the window. In other words, you can keep your end user from shrinking the window so much that its controls become invisible or unrecognizable.

Minimum Width
Specify the minimum width of the window in dialog units. Dialog units are based on the window's font and are 1/4 of the average character width.

Zero sets the window minimum to the size at which the window opens (not necessarily the design time size). In other words, it takes into account any .INI setting plus any runtime Property syntax. Thus, we allow the developer to open the window, perform any dynamic control production (including resizing the window) before the minimum restriction takes effect.

Minimum Height
Specify the minimum height of the window in dialog units. Dialog units are based on the window's font and are 1/8 of the character height.

Zero sets the window minimum to the size at which the window opens (not necessarily the design time size). In other words, it takes into account any .INI setting plus any runtime Property syntax. Thus, we allow the developer to open the window, perform any dynamic control production (including resizing the window) before the minimum restriction takes effect.

Restrict Maximum Window Size
Check this box to specify a maximum window height and width. This lets you enforce a maximum reasonable size of the window.

Maximum Width
Specify the maximum width of the window in dialog units. Dialog units are based on the window's font and are 1/4 of the average character width.

Zero sets the window maximum to the size at which the window opens (not necessarily the design time size). In other words, it takes into account any .INI setting plus any runtime Property syntax. Thus, we allow the developer to open the window, perform any dynamic control production (including resizing the window) before the maximum restriction takes effect.

Maximum Height
Specify the maximum height of the window in dialog units. Dialog units are based on the window's font and are 1/8 of the character height.

Zero sets the window maximum to the size at which the window opens (not necessarily the design time size). In other words, it takes into account any .INI setting plus any runtime Property syntax. Thus, we allow the developer to open the window, perform any dynamic control production (including resizing the window) before the maximum restriction takes effect.

Overriding the Resize Strategy for a Specific Control

Window Control
Select a control from the drop-down list.

Disable Resizing for this Control?
Check this box to prevent resizing of the selected control when the user resizes the window. The control will retain its design-time dimensions.

Horizontal Resize Strategy

Specify how the control's width is determined when the end user resizes the window.

Choose from:

Lock Width	The control's design time width does not change.
Constant Right Border	Locks right edge, moves left.

Vertical Resize Strategy

Specify how the control's height is determined when the end user resizes the window.

Choose from:

Lock Height	The control's design time height does not change.
Constant Bottom Border	Locks bottom edge, moves top.

Horizontal Positional Strategy

Specify how the control's horizontal position is determined when the end user resizes the window.

Choose from:

Move	The entire control moves in the direction of the parent’s horizontal resizing.
Lock Position	The control's left edge maintains a fixed distance (the design time distance) from parent's left edge.
Fix Right	The control's right edge maintains a proportional distance from parent's right edge.
Fix Left	The control's left edge maintains a proportional distance from parent's left edge.
Fix Center	The control's center maintains a proportional distance from parent's center.
Fix Nearest	Applies Fix Right or Fix Left, whichever is appropriate.
Fix To Center	The control's center maintains a proportional distance from it’s own center, not the parent’s center.

Vertical Positional Strategy

Specify how the control's vertical position is determined when the end user resizes the window.

Choose from:

Move	The entire control moves in the direction of the parent’s vertical resizing
Lock Position	The control's top edge maintains a fixed distance (the design time distance) from parent's top edge.
Fix Bottom	The control's bottom edge maintains a proportional distance from parent's bottom edge.
Fix Top	The control's top edge maintains a proportional distance from parent's top edge.
Fix Center	The control's center maintains a proportional distance from parent's center.

Fix Nearest	Applies Fix Top or Fix Bottom, whichever is appropriate.
Fix To Center	The control's center maintains a proportional distance from it's own center, not the parent's center.

Resizer Configuration Options

The **Classes** tab, located on the WindowResize template dialog, contains standard Classes options, with the following exceptions:

Auto-Find Parent Controls

Controls whether the SetParentDefaults method will be called during the Resize Initialization. The Default value is "Yes".

Optimize Moves

Sets the Resizer Class DeferMoves property. The default value is FALSE (No).

See the links to the appropriate methods above for more information.

Wise-Generate Wise Installation Script

This Extension template automatically generates a Wise Installation script for your application.

Script Path	Specify the path for the Install script. This is specific to this application
Overwrite with default .wse	Check this box to copy the default Wise Script (installed to your \TEMPLATE directory) to your application's local directory (overwriting the local script). Clear the check box if you want to maintain the local script file for this application.
Include External DLLs	Check this box if you want the runtime DLLs included in this installation script.
Application Title	Specify the name of your application as you want it to appear on the shortcut the installation creates.
Program Group Label	Specify the name of the program group you want the installation to create.
Default Folder	Specify the folder you want the installation to offer in its "Select Folder" step..
Additional Files to Install	Press this button to add other files (e.g., on-line documentation, data files, etc.). These files are added to the installation script. When adding files, you can also specify whether or not to create a shortcut for each file and the label to display for the item.

The Wise script this template creates does not list install files in the Wise for Clarion environment.

Business Rules Manager - Global Extension Settings

In Clarion terms, a *business rule* refers to your business data values stored through the application and to the constraints on attempted changes to those data values.

Business rules are precise statements that describe, constrain, and control selected data elements within your application.

The Global Extension allows you to define rules that can be used throughout your application. In each procedure, you can choose to override all, or part, of these business rules.

The Business Rules Manager template provides the following global prompts:

Mode:

Original/Clone selection

The Global Extension operates in one of two modes. In **Original** mode, all of the prompts described in this Help topic appear. When code is generated, the information entered is exported to a file. In **Clone** mode, only the **Mode** prompts appear. The extension imports rules information from the file.

This feature allows you to use the Global Extension in multiple-DLL projects without having to re-enter business rules for each DLL.

Rules File Name

Enter a name for the file, which is exported in **Original** mode and imported in **Clone** mode. The extension for this file will always be **.brf**.

All Rules:

The Business Rules Manager template provides to the user at runtime a display window containing a list of rules, and displays any errors made (rules broken) based on the data entered (or not entered).

Default Description

Enter a default description to use for the title bar of this window.

Error-indicator image

Enter an icon image to use that will identify rules errors to the user within the rules list.

Default distance from right

Enter a number in pixels that will separate the error indicator image from the specific rule description.

Global Rules

Press this button to access the Global Rules dialog

View Rules Icons

This group allows you to specify icons to be displayed in the Rules List dialog window. You can designate an icon for the **Header** (top element in a rules tree – normally the Default Description), **Valid Rule** (Rule names that are validated), and **Broken Rule** (Rule names that are not validated).

When any Rule is broken

Regardless of where the Rules Set is defined (Global or Local), the settings here designate specific controls that must be disabled or hidden anytime that a rule is broken. These settings can be overridden on the procedure level.

Disable Form "SaveButton"

Check this box to disable any SaveButton control template when a rule is broken.

Hide Browse "InsertButton"

Check this box to disable any InsertButton that is a populated component of the Browse Update control template when a rule is broken.

Hide Browse "ChangeButton"

Check this box to disable any ChangeButton that is a populated component of the Browse Update control template when a rule is broken.

Hide Browse "DeleteButton"

Check this box to disable any DeleteButton that is a populated component of the Browse Update control template when a rule is broken.

Note:

The specified controls above may be disabled or hidden when the window is first opened, after a rule check reveals one that is broken.

All Rules – Other Controls

Press this button to display a dialog that allows you to hide/unhide or enable/disable special controls anytime that a rule is broken.

Initialize User Rules

Check this box to allow inclusion of a set of rules that are read from an external file. This file must use a TopSpeed format, and can be defined internally or declared in a dictionary. If the file is declared in a dictionary, it must have 3 column names defined as follows:

```
Control       STRING(128)
Description   STRING(256)
Expression    STRING(512)
```

User rules file name

Enter the name of the Topspeed file to be used. This must be a string constant (no variables allowed). Example: *UserRules.TPS*

User rules file password

If the User rules file is encrypted, enter a valid password here.

Global Rules

The Global Rules dialog window allows you to define unlimited *sets* of Business Rules. Examples might be Global Rules, Customer Rules, Product Rules, Invoice Rules, etc.

Use the appropriate update buttons to add a new set of rules, modify an existing set, or delete an existing set. The following prompts are provided in the update dialog window:

Name

Enter a meaningful name to define your rules set. Example: *GlobalRules*, *CustomerRules*, etc. Note: No spaces are allowed in this name set.

Description

Enter a textual description that describes the rules set. Example: *Customer Information Rules*

In the Global Rules update dialog, you are provided with a list of the specific rules that are contained in the defined set. Use the update buttons to add, modify, or delete these specific rules. The following prompts are provided:

Rule Name

Enter a specific rule name here. The rule name must be in a variable format (e.g. – no spaces, colons and underscore characters permitted, etc.)

Rule Description

Enter a meaningful description of the specific rule that you have defined. Example: "This rule checks if the customer is domestic (CUS:Country = USA)"

Rule Definition – Evaluation Expression

Enter a valid expression that is used as your rules criteria. Example: CUS:Country <> ‘USA’, or CUS:AccountNumber <> 0.

The specific rule that you define will always return a zero if the expression you enter evaluates to "False".

Press the "**E**" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the prompt.

Field to link to

Press the ellipsis button to select a valid field name to link the specific rule to, which will be evaluated whenever this field is populated, or added to the Rules Hot Field list (found in the procedure based Local Rules template).

Display an error indicator when a rule is broken

Check this box to display an error indicator (message) when the specific rule is broken (fails).

Controls Button

Press the Controls button to open a dialog window used to identify a set of generic controls (identified by Field Equate Labels) which will be enabled or unhidden when the rule is evaluated to "True", or disabled or hidden when the rule is evaluated to "False".

Rules - Controls

Enable/Disable Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically enabled if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically disabled. Enter a Field Equate Label of the control that you expect to evaluate.

Hide/Unhide Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically unhidden if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically hidden. Enter a Field Equate Label of the control that you expect to evaluate.

Add OK and Cancel Controls

The Add OK and Cancel Controls button populates the appropriate list box with a predefined list of controls.

Local Business Rules Manager

The Local Business Rules Manager is used to designate the business rules that will be checked in the designated procedure. The Global Business Rules extension should first be applied before setting the following options:

All Rules Description

Enter the name to be used as the title text of the rules list window. The Business Rules Handler controls the rules list window.

Check Rules After Open Window

Check this box if you wish to check your selected business rules immediately after the window is opened. You can then give the user the option to view "broken rules", so that they are aware of information that must be entered.

Check Rules After Fields Change

Check this box if you wish to check your selected business rules immediately after fields are completed (accepted) on the window. This can allow your business rules to be consistently checked, especially in the scenario where multiple fields are used with a particular rule.

Check for Global Rules Controls

Check this box to apply the global settings of selected controls to apply to this procedure. See Business Rules Controls for more information.

Hot Fields

Press this button to designate special hot fields that may or may not be populated on your window, but are included in the business rules that you need to check.

Used Global Rules

Press this button to allow you to include rule bases (or sets) that you have defined globally.

Local Rules

Press this button to allow you to create new rule bases (or sets) to check locally in this procedure.

Override Global Form/Browse Controls Actions

Check this box to enable the **When any Rule is broken** group, which allows you to override the Global settings.

When any Rule is broken

Regardless of where the Rules Set is defined (Global or Local), the settings here designate specific controls that must be disabled or hidden anytime that a rule is broken. These settings can be overridden on the procedure level.

Disable Form "SaveButton"
Check this box to disable any SaveButton control template when a rule is broken.

Hide Browse "InsertButton"
Check this box to disable any InsertButton that is a populated component of the Browse Update control template when a rule is broken.

Hide Browse "ChangeButton"
Check this box to disable any ChangeButton that is a populated component of the Browse Update control template when a rule is broken.

Hide Browse "DeleteButton"
Check this box to disable any DeleteButton that is a populated component of the Browse Update control template when a rule is broken.

The specified controls above may be disabled or hidden when the window is first opened, after a rule check reveals one that is broken.

Check for Global All Rules – Other Controls

Check this box to allow this procedure to check for global control equates to hide/unhide or disable/enable when a rule is broken. When unchecked, the global settings are ignored by this procedure.

All Rules – Other Controls

Press this button to display a dialog that allows you to hide/unhide or enable/disable special controls anytime that a rule is broken. On the procedure level, the controls you set are only valid within this procedure's scope.

Local Business Rules

The Local Rules dialog window allows you to define unlimited *sets* of Business Rules. Examples might be Customer Rules, Product Rules, Invoice Rules, etc.

Use the appropriate update buttons to add a new set of rules, modify an existing set, or delete an existing set. The following prompts are provided in the update dialog window:

Name

Enter a meaningful name to define your rules set. Example: *CustomerRules*, etc. Note: No spaces are allowed in this name set.

Description

Enter a textual description that describes the rules set. Example: *Customer Information Rules*

In the Local Rules update dialog, you are provided with a list of the specific rules that are contained in the defined set. Use the update buttons to add, modify, or delete these specific rules.

Local Business Rules - Hot Fields

The Hot Fields dialog window allows you to enter additional fields that may not be populated on the procedure's window, but are a part of some or all of the selected business rules, and need to be included.

Press the Insert button to add a Hot Field, and on the subsequent dialog, press the ellipsis to select a hot field from the Field Selection List.

Local Business Rules - Locally Defined

In the Local Rules update dialog, you are provided with a list of the specific rules that are contained in the defined set. Use the update buttons to add, modify, or delete these specific rules. The following prompts are provided:

Rule Name

Enter a specific rule name here. The rule name must be in a variable format (e.g. – no spaces, colons and underscore characters permitted, etc.)

Rule Description

Enter a meaningful description of the specific rule that you have defined. Example: "This rule checks if the customer is domestic (CUS:Country = USA)"

Rule Definition – Evaluation Expression

Enter a valid expression that is used as your rules criteria. Example: CUS:Country <> ‘USA’, or CUS:AccountNumber <> 0.

The specific rule that you define will always return a zero if the expression you enter evaluates to "False".

Field to link to

Press the ellipsis button to select a valid field name to link the specific rule to. Whenever this field is populated or added to the Rules Hot Field list (found in the procedure based Local Rules template)

Display an error indicator when a rule is broken

Check this box to display an error indicator (message) when the specific rule is broken (fails).

Controls Button

Press the Controls button to open a dialog window used to identify a set of generic controls (identified by Field Equate Labels) which will be enabled or unhidden when the rule is evaluated to "True", or disabled or hidden when the rule is evaluated to "False".

Local Business Rules - Special Controls

Enable/Disable Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically enabled if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically disabled. Enter a Field Equate Label of the control that you expect to evaluate.

Hide/Unhide Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically unhidden if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically hidden. Enter a Field Equate Label of the control that you expect to evaluate.

Add OK and Cancel Controls

The Add OK and Cancel Controls button populates the appropriate list box with a predefined list of controls.

Local Business Rules - Used Controls

Enable/Disable Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically enabled if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically disabled. Enter a Field Equate Label of the control that you expect to evaluate.

Hide/Unhide Controls

Press the update buttons to add, change, or remove, selected controls that will be automatically unhidden if the specific rule that you have defined evaluates to a "True" condition. If the specific rule defined evaluates to a "False" condition, the controls in the list box will be automatically hidden. Enter a Field Equate Label of the control that you expect to evaluate.

Add OK and Cancel Controls

The Add OK and Cancel Controls button populates the appropriate list box with a predefined list of controls.

Local Business Rules - Used Global Rules

The Used Global Rules dialog window provide a list of Global Business Rule sets (bases) that you have previously defined in the Business Rules Global Extension.

Select a rulebase name from the list, and press the **Properties** button. The following prompts are available:

Disable this rule?

Check this box to disable the selected global rulebase from the active procedure. Rules that are disabled will not have an icon in the list that identifies them as selected.

Override Display Indicator?

Check this box to override the rulebase display indicator that was defined in the Global Business Rules template

Display an error indicator when a rule is broken

Check this box to display an error indicator (message) when the specific rule is broken (fails).

Use default control if found?

Check this box to use a default control on the window that equates to the current rule defined. If a default control is not found, you can uncheck this box and select one from the subsequent drop list.

Default distance from right

Enter a number in pixels that will separate the error indicator image from the specific rule description.

Controls Button

Press the Controls button to open a dialog window used to identify a set of generic controls (identified by Field Equate Labels) which will be enabled or unhidden when the rule is evaluated to "True", or disabled or hidden when the rule is evaluated to "False".

Code Templates

Select Embed Type

This dialog lets you specify what to embed at this point. CLICK on the item to embed, then press the **Select** button.

The choices are:

Call a procedure	Provides a dialog that lets you specify a procedure to call.
SOURCE	Calls the Text Editor, allowing you to hand code the embedded source.
Code Templates	The templates listed here depend on the templates you have registered. Code templates generate executable code. Generally, each Code template has one well-defined task. For example, the Initiate Thread Code template simply starts a new execution thread, and no more. Typically, the Code template provides a dialog box with options and instructions.

There are many Code Templates included with Clarion (see the help TOC for a complete list):

CallABCMethod	Generates code to call an ABC Library object met
Call Procedure As Lookup	This code template lets you call a procedure, usually a Browse, with a request to make a selection.
Close Current Window	This code template simply posts an EVENT:CloseWindow, which tells the currently active window to close.
Control Value Validation	This code template validates the value of an entry control (ENTRY, LIST, COMBO, or SPIN). You can add this code template to a field event on a control; at the Accepted or Selected embed point.
DisplayPopupMenu	Generates code to define and display a popup menu, and optionally, act on the end user's selection.
Export to XML	Used to write the contents of a valid data source (or origin) to an external XML file.
Initiate Thread	This code template initiates an execution thread when opening an MDI window.

Lookup Up Non-Related Record	This code template is used to perform a lookup of a value based on a relationship not defined in the Data Dictionary (ad hoc relations). You can add this code template to the Lookup Up Related Records embed point.
ResizeSetStrategy	Lets you override the default resize strategy for a particular control
SelectToolbarTarget	This Code template provides an easy way to control which Browsebox is tied to the Toolbar navigation buttons (see **FrameBrowseControl**).
SetABCProperty	Generates code to set a public property of an ABC Library object
SetProperty	This Code template provides an easy way to set a runtime property of any control on a window.

ADO Browse Refresh Code Template

The ADO Browse Refresh Code Template allows you to refresh an ADO Browse contents at nearly any source embed point. This is useful when a data value is updated, and its value directly affects the browse box contents.

The code template has no prompts, and simply calls the ADO object's Refresh method.

ADO Browse to XML Code Template

The ADO Browse to XML Code Template allows you to export the contents of an active browse box to an external XML file. The appropriate XML tags are automatically added to the file.

Press the ellipsis to select a variable that will hold the XML external filename. The variable must hold the full path and filename. If the path is omitted, the current program directory is used.

CallABCMethod

The CallABCMethod template generates code to call an ABC Library object method. This template generates code similar to the following:

```
Default:City = INIMgr.Fetch('Preferences','City')
```

Object Name
Select the label of the object from the list. The list contains all ABC compliant objects in scope for this procedure.

Method to Call
Select the method to call from the drop-down list. Scroll the list horizontally or press the **Application Builder Class Viewer** button to see all the method parameters and return values.

Passed Parameters
Type the parameter list to pass. Enclose the parameters in parenthese and separate them with commas. The parameters may be literal values, expressions, or variable names. Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Passed Parameters** prompt.

Return Value Assignment
Type the variable to receive the called method's return value. This field is only available for methods that return a value.

Application Builder Class Viewer
Press this button to display classes, properties, and methods used by the ABC Templates, and the relationships between parent and derived (child) classes. This utility can help you analyze and understand the classes that the ABC Templates use.

Call a Procedure (Extended) code template

This code template is similar to the actions available in the Menu Editor , and adds extended features through the following template prompts:

Action Choose *Call a Procedure* or *Run a Program* from the drop list provided.

The remaining prompts are similar to those found in the Menu Editor for each menu item's actions, and are shown below:

Procedure Definition Prompts

Procedure Name From the **Procedure Name** drop down list, choose an existing procedure name, or type a new procedure name. A new procedure appears as a "ToDo" item in your Application Tree.

Initiate a Thread Optionally check the **Initiate a Thread** box. If the procedure initiates a thread, specify the Thread Stack size. Clarion uses the START function to initiate a new execution thread. You can optionally specify **Parameters**, **Requested File Action**, or both.

Thread Stack Accept the default value in the **Thread Stack** spin box unless you have extraordinary program requirements. To change the value, type in a new value or click on the spin box arrows.

Parameters In the **Parameters** field, optionally type a list of variables or data structures passed to the procedure. Press the "**E**" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Parameters** prompt.

Return Thread Variable If the thread procedure called is prototyped to return a value, press the ellipsis button here to select a variable to receive the returned value.

Requested File Action From the drop down list, optionally select ***None, Insert, Change, Delete***, or ***Select***. The default selection is ***None***. The Global Request variable gets the selected value. The called procedure can then check the value of the Global Request variable and perform the requested file action.

Return Value Variable If the procedure called is prototyped to return a value, press the ellipsis button here to select a variable to receive the returned value.

Optionally check the **Reference Assign** check box if the variable defined is a reference variable.

Program Definition Prompts

Program Name	Type the program name. The program name must be in your path or current folder, else enter the full path and executable program. Quotes are added to your entry so you don't need to enter any.
Parameters	Optionally type a list of values that are passed to the program. Press the "**E**" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Parameters** prompt.

Call Procedure As Lookup code template

This Code template calls a procedure to select a record. It sets a variable called RequestCompleted to advise whether the lookup was successful.

Lookup Procedure	Specifies the procedure to call.
Code before	Type in any executable code to execute before performing the lookup. Multiple statements can be used if separated by a semicolon.
Code After, Completed	Type in any executable code to execute after completing a lookup. Multiple statements can be used if separated by a semicolon.
Code After, Canceled	Type in any executable code to execute if the lookup is canceled. Multiple statements can be used if separated by a semicolon.

Press the **E** button where appropriate to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompts.

Close Current Window code template

This code template simply posts an EVENT:CloseWindow, which tells the currently active window to close. There are no prompts to fill in.

Control Value Validation code template

This code template gets the value of the control and matches it against the value in the key. You can add this code template on an ENTRY, SPIN, LIST, or COMBO control; at the Accepted or Selected embed point. The code generated by this code template gets the value in the control, then matches it against the value in the key.

It can also call a lookup procedure to let the end user select a value. You can check whether the end user has successfully completed the lookup procedure by checking the value of the ThisWindow.Response (or SELF.Response) variable.

See Also: Request and Response

DisplayPopupMenu

The DisplayPopupMenu template generates code to define and display a popup menu, and optionally, act on the end user's selection. You can set the popup menu items to mimic existing buttons on the window so that the associated menu item text matches the *button* text, is enabled only when the *button* is enabled, and, when selected, invokes the *button* action. The DisplayPopupMenu template relies on the PopupClass to accomplish its tasks.

String variable for Press the ellipsis (...) button to select or define a string variable to receive the end user's popup menu selection. After the popup menu displays, this variable contains the selected item's text minus any special characters. That is, the variable contains only characters 'A-Z', 'a-z', and '0-9'. If the resulting value is not unique for the menu, the PopupClass appends a sequence number to the value to make it unique.

You may interrogate this variable and perform actions depending on its value. If you rely on the PopupClass mimic capability to perform appropriate actions, then you can leave this field blank. See *Item Properties* for more information on mimic.

Build Menu From Choose how the popup menu and its items are defined. Choose from:

Menu String Use the **Menu String** field to type the menu definition, then use the **Item Properties** to define each item's behavior.

Item List Use the **Menu Items** button to define menu items one at a time.

INI File Use the **Menu Description** field to name the INI file section which contains the menu definition. By default, the template code uses the global INIMgr object declared by the ABC Application template. If you have not specified an INI file to use, the INIMgr object uses Windows INI file. See *Template Overview—Global Options Tab*.

Menu Description Type the INI file section which contains the menu definition.

Menu String Type a menu definition string. The *Language Reference* describes the syntax for the menu definition string under the *selections* parameter for the POPUP command.

Item Properties Press this button to define the properties for each popup menu item. Only items specified in the Menu String are valid. You can set the popup menu items to mimic existing buttons on the window so that the associated menu item text matches the *button* text, is enabled only when the *button* is enabled, and, when selected, invokes the *button* action. You can also set the popup menu items to post an event to a control.

Menu Items Press this button to define the text for each popup menu item. You can set the popup menu items to mimic existing buttons on the window so that the associated menu item text matches the *button* text, is enabled only when the *button* is enabled, and, when selected, invokes the *button* action. You can also set the popup menu items to post an event to a control.

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

Export to XML code template

The Export to XML code template is used to write the contents of a valid data source (or origin) to an external XML file. This code template is actually a wrapper around the built-in class library support for XML parsing.

The following prompts are provided:

Data Origin Press the ellipsis button to select a valid label of a GROUP, FILE, VIEW or QUEUE that holds the XML contents to export to an XML file. Press the "**E**" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Data Origin** prompt.

Mapping File Press the ellipsis button to select a valid XML mapping file to use during the export process. In Clarion terms, mapping refers to associating an XML tag or schema to a database column name. If you wish to select this file at runtime, check the **Select mapping file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Export File Press the ellipsis button to select the valid XML source file name to export to. If you wish to select this file at runtime, check the **Select export file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Root Tag: Enter a hard coded string value, or specify a variable that will identify the root tag name (using the *!variablename* format) to begin the export to.

Row Tag: Enter a hard coded string value, or specify a variable that will identify the row tag name (using the *!variablename* format) to begin the export to.

Root and Row Tags can contain letters, numbers, and other characters. They must not start with a number or punctuation character or with the letters xml (or XML or Xml ..). They cannot contain spaces

XML Style Select *Tag-based* or *Attribute-based*. This setting controls the XML formatted output style.

For more information, please review the *Clarion XML Support Reference Guide* PDF file included in your install.

FromXML code template

This code template is used to import an XML file's contents into a designated target structure. It is similar to the Import From XML code template, but provides additional settings for more control as to the input contents.

Select Structure

Press the ellipsis button to select a valid label of a GROUP, FILE, VIEW or QUEUE to hold the contents of the import XML file.

Import file

Press the ellipsis button to select a valid XML source file to import. If you wish to select this file name at runtime, check the **Select import file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Mapping file (optional)

Press the ellipsis button to select a valid XML mapping file to use during import. In Clarion terms, mapping refers to associating an XML tag or schema to a database column name. If you wish to select this file at runtime, check the **Select mapping file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Root Tag:

Enter a hard coded string value, or specify a variable that will identify the root tag name (using the *!variablename* format) to begin the import from.

Row Tag:

Enter a hard coded string value, or specify a variable that will identify the row tag name (using the *!variablename* format) to begin the import from.

Use Schema on import

There are two ways to describe the elements of an XML document. The default is to use a DTD (Document Type Definition) to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. The alternative method is to use an XML Schema. Check this box to designate the Schema on import instead of the DTD.

Perform a silent import

Check this box if you wish to suppress error generation during the import process.

Customize Columns

Press this button that displays a list box interface where you can add custom mapping from the XML Tag Name to an existing column in your database.

Column

Select a valid label from the drop list. The elements contained here are reads from the **Select Structure** value.

XML Tag Name

Enter the target tag name to match to the selected column.

Picture

Enter a default picture to use to format the imported XML data.

XML Data Format

Select from *Text*, *Cdata*, or *Base64*. The default is *Text*.

XML Style

Select *Tag-based* or *Attribute-based*. This setting controls the XML formatted output style.

GetFullDragSetting code template

This code template provides an easy way to detect the full window area dragging mode in your runtime application.

Target Variable Press the ellipsis button to select a variable name that will store the value of the FULLDRAG() API. This variable should be a numeric type.

Source Equivalence: `Target Variable = FULLDRAG()`

GetOSVersion code template

This code template provides an easy way to detect the operating system where your application is running by using the new PROP:WindowsVersion runtime property.

Target Variable Press the ellipsis button to select a variable name that will store the value of the runtime property. This variable should be a STRING or CSTRING type.

Source Equivalence:

<*Target Variable*> = SYSTEM{PROP:WindowsVersion}

Import From XML code template

The Import From XML code template is used to read the contents of an external XML into a valid target destination. This code template is actually a wrapper around the built-in class library support for XML parsing.

The following prompts are provided:

Import File — Press the ellipsis button to select a valid XML source file to import. If you wish to select this file name at runtime, check the **Select import file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Mapping File — Press the ellipsis button to select a valid XML mapping file to use during import. In Clarion terms, mapping refers to associating an XML tag or schema to a database column name. If you wish to select this file at runtime, check the **Select mapping file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Destination — Press the ellipsis button to select a valid label of a GROUP, FILE, VIEW or QUEUE to hold the contents of the import XML file.

XML Style — Select *Tag-based* or *Attribute-based*. This setting controls the XML formatted output style.

Example – Tag based output:

```
<?xml version="1.0" ?>
- <dataroot>
- <Table>
  <id>1</id>
  <firstname>Fred</firstname>
  <lastname>Flintstone</lastname>
  <gender>M</gender>
  </Table>
```

Example – Attribute based output:

```
<?xml version="1.0" ?>
- <dataroot>
  <row id="1" firstname="Fred" lastname="Flintstone" gender="M" />
  <row id="2" firstname="Andrew" lastname="Guidroz" gender="M" />
  <row id="4" firstname="Gavin" lastname="Holiday" gender="M" />

  <row id="35" firstname="Hello" lastname="There" gender="f" />
  </dataroot>
```

For more information, please review the *Clarion XML Support Reference Guide* PDF file included in your install.

InitiateThread code template

When opening an MDI window from an Application Frame, you must initiate an execution thread. This Code template provides an easy way to initiate a thread.

When you START a procedure on its own thread, the procedure and its window operate independently of other threads in the same program; that is, the end user can switch focus between each execution thread at will. These are "**modeless**" windows.

If you don't initiate a new thread, the program behavior depends on whether the procedure's window has the MDI attribute. *A non-MDI* child window on the same thread as its parent, blocks access to all other threads in the program. This is an "**application modal**" window. When the application modal window closes, the other execution threads are available again. *An MDI* child window on the same thread as its parent, blocks access only to its parent window. When the MDI child window closes, its parent window regains focus.

In the **Prompts for Initiate Thread** dialog, simply name the procedure that opens the MDI window. Optionally, you can modify the size of the stack to allocate to the new execution thread. The default stack is 25,000 bytes.

You can optionally add a line of code to execute if the application was unable to open the thread. Type in the edit box labelled **Error Handling**. For example,

MESSAGE('Could not Start Thread','Error',ICON:HAND)

would display a message box with the halt (hand) icon, if the thread failed to start.

You can add a procedure name to call upon an error by typing the name of the procedure in the **Error Handling** box. You would then add the procedure to the **Application Tree** with the **Insert Procedure** command.

Lookup Non-Related Record code template

The LookupNonRelatedRecord template is used to perform a lookup of a value based on a relationship, whether it is or is not defined in the data dictionary (Ad hoc relation). You can add this Code template to the Lookup Up Related Records embed point.

Lookup Key Type in the key name or press the ellipsis (...) button to select the key from the **Select Key** dialog.

The lookup key is used to perform the lookup into the lookup file. This *must* be a unique key. If the key is a multi-component key, the other key elements must be primed before executing this Code template.

Lookup Field Type in the field name or press the ellipsis (...) button to select the field from the Component list.

The Lookup Field must be a component of the Lookup Key. This is the unique value within the lookup file.

Related Field Type in the related field or press the ellipsis (...) button to select it from the **Select Column** dialog.

The Related Field provides the unique value used to perform the lookup.

This template generates code similar to the following:

```
ST:StateCode = CUST:State       ! Move value for lookup
Access:State.Fetch(ST:ByCode)   ! Get value from file
```

Process Transaction Frame Checkpoint code template

This code template requires the ProcessTransactionFrame extension template.

It is used to generate an even more precise control of the **number of records** that will be included in each transaction frame. You must have the **Generate one transaction for each record read** check box active in the ProcessTransactionFrame extension.

ResizeSetStrategy

The ResizeSetStrategy template lets you override the default resize strategy for a particular control. It is designed exclusively for the **Set resize strategy** embed point for a specific control. See WindowResize extension template for more information on the default resize strategies.

Insert the code template at the **Set resize strategy** embed point for the control for which to set the resize strategy, then complete the following prompts.

Horizontal Resize Strategy	Specify how the control's width is determined when the end user resizes the window. Choose from:
Lock Width	The control's design time width does not change.
Constant Right Border	Locks right edge, moves left.
Vertical Resize Strategy	Specify how the control's height is determined when the end user resizes the window. Choose from:
Lock Height	The control's design time height does not change.
Constant Bottom Border	Locks bottom edge, moves top.
Horizontal Positional Strategy	Specify how the control's horizontal position is determined when the end user resizes the window. Choose from:
Lock Position	The control's left edge maintains a fixed distance (the design time distance) from parent's left edge.
Fix Right	The control's right edge maintains a proportional distance from parent's right edge.
Fix Left	The control's left edge maintains a proportional distance from parent's left edge.
Fix Center	The control's center maintains a proportional distance from parent's center.
Fix Nearest	Applies Fix Right or Fix Left, whichever is appropriate.
Vertical Positional Strategy	Specify how the control's vertical position is determined when the end user resizes the window. Choose from:
Lock Position	The control's top edge maintains a fixed distance (the design

	time distance) from parent's top edge.
Fix Bottom	The control's bottom edge maintains a proportional distance from parent's bottom edge.
Fix Top	The control's top edge maintains a proportional distance from parent's top edge.
Fix Center	The control's center maintains a proportional distance from parent's center.
Fix Nearest	Applies Fix Top or Fix Bottom, whichever is appropriate.

SelectToolbarTarget code template

This Code template provides an easy way for developers to control which Browsebox in a given procedure is tied to the Toolbar navigation buttons (see *FrameBrowseControl*).

Toolbar Navigation Target

Select the Browsebox or the Form that is controlled by the FrameBrowseControl navigation buttons.

This Code template generates the following code:

```
DO BRW1::AssignButtons
```

…where *BRW1* identifies the Browsebox to navigate, or

```
DO FORM::AssignButtons
```

…to tie the navigation buttons to the active Form procedure (that is, the SaveButton Control template).

SetABCProperty

The SetABCProperty template generates code to set a public property of an ABC Library object. This template generates code similar to the following:

```
BRW2.ActiveInvisible = True
```

Object Name
Select the label of the object from the list. The list contains all ABC compliant objects in scope for this procedure.

Property to Set
Select the property to set from the drop-down list. See *ABC Library* for more information on these properties.

Value to Set
Type a variable, constant, or valid Clarion expression to assign to the property.

Assign as Reference?
Check this box to generate a reference assignment (*object.property &= value*). Clear the box to generate a simple assignment (*object.property = value*). See *Reference Assignments* in the *Language Reference* for more information.

SetFullDragSetting code template

This code template provides an easy way to enable full window area dragging in your application.

Turn On Check this box to enable full window dragging in your application.

Use this code template to force full window dragging if you want to force this look and feel in your applications, or a user's operating system is having problems with the Enable Window Frame Dragging setting in the application's Global Properties.

The best place to include this template is any embed point that is available prior to opening the Application Frame.

Source Equivalence: **FullDrag(1)**

SetProperty code template

This Code template provides an easy way to set a runtime property of any control on a window.

Control Select the field equate label for one of the window controls from the drop down list.

Property Select the runtime property to set from the drop down list.

Value The label of a variable, a constant, or an expression to assign to the selected runtime property.

This Code template generates the following code:

?MyControl{PROP:Whatever} = value

ToXML code template

This code template is used to export contents to an XML file's from the designated data origin.

Data Origin

Press the ellipsis button to select a valid label of a GROUP, FILE, VIEW or QUEUE that holds the XML contents to export to an XML file. Press the "**E**" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Data Origin** prompt.

Export file

Press the ellipsis button to select the valid XML source file name to export to. If you wish to select this file at runtime, check the **Select export file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Mapping file (optional)

Press the ellipsis button to select a valid XML mapping file to use during the export process. In Clarion terms, mapping refers to associating an XML tag or schema to a database column name. If you wish to select this file at runtime, check the **Select mapping file at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Root Tag:

Enter a hard coded string value, or specify a variable that will identify the root tag name (using the *!variablename* format) to begin the import from.

Row Tag:

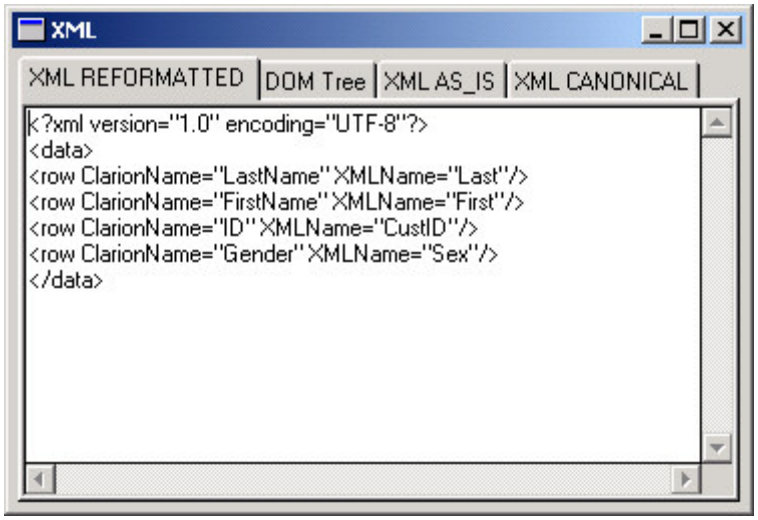
Enter a hard coded string value, or specify a variable that will identify the row tag name (using the *!variablename* format) to begin the import from.

XML Style

Select *Tag-based* or *Attribute-based*. This setting controls the XML formatted output style.

ViewXML code template

The ViewXML Code template is used to analyze an XML file with the use of a default viewer provided by an internal class method. The following window, with four different views, is displayed at runtime:



XML file to view

Press the ellipsis button to select the valid XML source file name to view. If you wish to select this file at runtime, check the **Select file to view at runtime** check box. You may also specify a variable here (Example: *!FilenameVariable*).

Control Templates

Select Control Template

This dialog lets you choose a control template to add a control plus its supporting source code to your procedure.

CLICK on a control template from the list, then press the **Select** button.

If you check the **Translate controls to control templates when populating** box on the **Application Properties** dialog, the list contains control templates and controls without templates. This is primarily a reminder that control templates with their built-in functionality are available as an alternative to do-it-yourself controls.

If you add third party, or your own customized templates to the Template Registry, they appear in the list. The following lists the control templates which ship with the Clarion ABC/Clarion templates:

ASCII View control template	This control template adds a list box in which you can display an ASCII (text) file.
ASCII Print Button	This control template adds a button to print an ASCII (text) file.
ASCII Search Button	This control template adds two buttons (Find and Find Next) to search an ASCII (text) file using object oriented code.
Browse Box	This control template places a LIST control in a window.
Browse Fuzzy Matching	The BrowseFuzzyMatching template adds a set of controls that lets end users perform a weighted search on the data in a BrowseBox. The controls include an entry field, a search button, a clear search button, and a group control that surrounds them.
BrowseGrid	The BrowseGrid template provides a group control used to define an area (within a BrowseBox) to display controls as a matrix.
BrowseNoRecords Button	This template populates a button that becomes enabled only when there are records displayed in a related Browse Box.
BrowsePrint Button	This template populates a button that selects a single record for you to print or process.
Browse QBE Button	Populates a button that calls a designated List or Form query interface.
BrowseQBEList	Populates a LIST and other associated controls that allows advanced queries to be created and stored.
Browse Select Button	This control template provides a quick way to return a value from a list box called to request a record.
BrowseToolbarControl	This control template places thirteen (13) standard command buttons on the toolbar of a Browse Procedure. When the user presses these buttons, the template generated code posts appropriate events (scroll up, scroll down, add, change, delete, help, etc.).

BrowseToolbox Button	This control template populates a button that calls a popup toolbox that allows access to common database activities (Insert, Change, Query, etc.).
Browse Update buttons	This control template provides a quick way to manage records in a list box.
BrowseViewButton	The BrowseViewButton template provides a **View** button to let the end user call the browse update procedure without update functionality.
CalendarButton	This template populates a lookup button that calls a popup calendar.
Cancel Button	This control template provides a convenient way to close a browse procedure and cancel a record request.
Close Button	This control template adds a single button control marked **Close** which closes down the current window.
DOS File Lookup	This control template adds an ellipsis (...) button which leads the end user to a standard Open File dialog.
Dynamic Image	The DynamicImage control provides an image control that must be placed within an existing BrowseGrid control.
Field Lookup Button	This control template adds an ellipsis (...) button to call a lookup procedure specified for an entry control.
File Drop	This control template adds a drop down list showing the contents of a selected field of a file listed in the data dictionary.
File Drop Combo	This control template adds a Combo Box with drop down list showing the contents of a selected field of a file listed in the data dictionary. It also allows updates to the file.
FormVCRButtons	This template populates a set of buttons that allow navigation and updates of a selected table.
FrameBrowseControl	This control template places thirteen (13) standard command buttons on the toolbar of an MDI APPLICATION (*Frame*). When the user presses these buttons, the template generated code posts appropriate events (scroll up, scroll down, add, change, delete, help, etc.) to the active procedure.
Help Button	This control template populates a button that allows you to call the help topic created for this window.
OLE Control	This control template adds an OLE control plus callback functions if the OLE contains an OCX.

PauseButton	This control template adds a button to allow pausing and restarting of a Process or Report procedure.
Populate control without control template	This is a do-it-yourself control, not a control template.
ProcessReportQBEButton	The ProcessReportQBEButton template provides a **Query** button to let the end user apply a dynamic (run-time) filter to a report.
RelationTree	This control template places a LIST control formatted as a tree in a window.
Relation Tree Update Buttons	Adds buttons to a window which call the appropriate update procedures for levels in a Relation Tree.
Relation Tree Expand/Contract Buttons	Adds buttons to a window which expand and contract all the levels in a Relation Tree.
ReportDateStamp	Adds two STRING controls to a REPORT: a "Report Date:" text STRING, and a formatted variable STRING to display the date.
ReportPageNumber	Adds a variable STRING to display the page number.
ReportTimeStamp	Adds two STRING controls to a REPORT: a "Report Time:" text STRING, and a formatted variable STRING to display the time.
RTFControl	The RTFControl template adds a TEXT control to a window. The necessary code is generated and used to manipulate the text within the control as Rich Text. This control has been superceded by the more powerful RTFText Control template.
RTFStatusBar	Creates a group of controls that display status information of a related RTFText Control.
RTFTextControl	Adds a TEXT control that manipulates the text within the control as Rich Text.
RTFToolbar	Adds a set of buttons that provides a wide variety of functions for a related RichText Control.
Save Button	This control template adds an **OK** button to close a window and save the action.
SaveChange Button	This template is normally used along with the FormVCR control template to allow writes of a record’s contents to a data file.
SortOrder Button	This template populates a button that provides a popup menu of browse sort order choices.
SortOrder Drop	This template populates a drop list that provides a list of browse

sort order choices.

ADO Browse Box Control Template

Click on a TAB above to jump to that area

The ADO Browse Box Control template provides the extensive features of the standard Browse Box control, and managed through an ADO connection layer.

In addition, there are new features that extend the functionality of the standard Browse Box, including:

- Ability to change column position at runtime, by dragging the appropriate column to its new position.
- Ability to click on a column header and sort by that column at runtime. You can also SHIFT click on an additional header in order to add another column to the sort order sequence. SHIFT click again to change the order from ascending to descending.
- Ability to HIDE/UNHIDE columns by pressing CTRL+RIGHTCLICK in the data area of the ADO Browse Box at runtime.
- More powerful template-based filtering options.

The following prompts are available:

General

Loading Method

Select **Page** or **Table** from the drop list provided. **Page** tells the Browse Box to return a page of records to the client when requested. Table returns all records of the primary table used in the Browse Box, unless limited below.

Limit the result set rows number

Check this box to limit the number of rows returned by the client's request.

Maximum rows

Specify a maximum number of rows to return. This prompt is only available if the **Limit the result set rows number** check box is checked.

Generate initial call to Refresh

Check this box to force the browse to immediately refresh all data elements when the browse is first loaded. This will allow any secondary elements to be updated properly.

Connection Group:

Connection Object

Choose a Connection object name from the drop list provided. This list should contain the connection object you created in the ADO Global support template.

Use a New Connection

If you do not wish to use any of the Global Connections that are available, check this box to create a new connection specifically for this browse box.

New Connection

Press this button to call the Connection Builder. On the subsequent dialog, enter a new name to use for the new **Connection Object**, and press the **Connection String Settings** button to access the Connection String Settings dialog.

DB Interface Group:

DB Interface object

The Database Interface object name is generated by the ADO template. If you need to override the default object name, enter a new name here.

CRecordset Attributes

Press the **Crecordset Attributes** button to access special properties that affects the ADO recordset. Each attribute is described in detail in the ADO Reference provided by Microsoft.

Cursor Location

The **CursorLocation** property on a **Connection** object or **Recordset** object indicates the location of the cursor engine.

Cursor Type

The **CursorType** property on a **Recordset** object indicates the type of the cursor engine.

LockType

The **LockType** property is used to set the type of locks placed on records that the provider should use when opening the **Recordset**.

Command Type

The **CommandType** property on a **Recordset** object Indicates the type of a **Command** object.

Execute Options

The **Execute** method on a **Connection** object executes the statement specified in the **CommandText** property

Table Mapper – BASETABLENAME is not supported

Check this box if your ADO data source does not support BASETABLENAME capability.

In ADO, when you have created a SELECT statement through your template design that contains the same column name, but coming from a different table, the fields' collection will have field objects with the same name.

For example,

```
SELECT Customer.SysID, order.SysId etc…
```

The Fields collection above will have 2 Field objects with the name *SysID*. In order to make sure that the values from those 2 fields will go into the right application variable, we need to know which table the field belongs to. There is a property in the Field properties collection called ***BASETABLENAME*** that provide this value.

Data Columns/Hot Fields

The Data Columns and Hot Fields tab control provide prompts that allow control of the data elements that will appear in the Browse Box, in the scrollable (Data Columns) and non-scrollable (Hot Fields) areas.

The following prompts are presented:

Automatically manage repeated columns name

Check this box to allow the ADO template to handle duplicate column names automatically. If you are writing custom SQL statements, and wish to name your own repeated columns, turn off this check box.

Data Columns

The Data Columns list box displays data elements that have been populated via the List Box Formatter. Press the **Properties** button to access the following prompts:

Query Field

Identifies a column as a field that can be queried (searched).

Column is a

Identifies the data element as one that is read directly from the ADO data source (**Table Column**), or one that is a variable defined within the application, like a computed or conditional field (**Expression**).

Use AS

Check this box if the column name is long and verbose, and you need to rename the column to a more descriptive and compacted name. This is useful if you are constructing long or complex SQL statements. This is an option for Table Columns, but required for all Expressions.

Unique Field ID/AS

Enter a unique Field ID to use with the **Use AS** option for Table Columns, or the AS option for an Expression..

Expression

Enter a valid expression to use for the defined Data Column

This column cannot be hidden

Check this box to restrict this Data Column from being hidden as set in the Default Behavior tab.

This column cannot be sorted

Check this box to restrict this Data Column from being a sort field as defined in the Default Behavior tab.

This Column is the Default sort column

Check this box to indicate that this column is to be used as the default sort. This will generate a call to the cBrowse method *ApplySort* with the corresponding column number in order to perform the sort and update of the list box header accordingly.

Use the **This Column is the Default sort column** check box to set the column to use as the default locator.

Hot Fields

The Hot Fields list box displays data elements that have been populated *outside* of the List Box Formatter, or data elements that may not be populated, but need to be referenced or updated in the procedure source. Hot fields are normally data elements that are related to the contents of the list box (i.e., Address information, text based descriptions, etc.). Press the **Properties** button to access the following prompts:

Hot Field

Press the ellipsis button to select a field to use as the hot field.

Column is a

Identifies the data element as one that is read directly from the ADO data source (**Table Column**), or one that is a variable defined within the application, like a computed or conditional field (**Expression**).

Use AS

Check this box if the column name is long and verbose, and you need to rename the column to a more descriptive and compacted name. This is useful if you are constructing long or complex SQL statements. This is an option for Table Columns, but required for all Expressions.

Unique Field ID/AS

Enter a unique Field ID to use with the **Use AS** option for Table Columns, or the AS option for an Expression..

Expression

Enter a valid expression to use for the defined Data Column

Expression Data Type

Select a valid data type from the drop list that the ADO layer will translate.

Default SQL

Because the SQL to access the data is vitally important, the ADO templates allow you to view the generated SELECT statement and customize it if necessary. This also provides a convenient way to customize the browse ordering, should you wish.

Regenerate SQL

Press this button to reset the original template-constructed SQL statements. This is useful should you need to start from scratch again before customizing your statements. Only enabled when the next prompt is active (checked).

Override SELECT SQL

Check this box to bypass the auto generation of the SELECT statement by the templates. Your custom statement will be substituted in its place. This box also enables the **Regenerate SQL** button, should you wish to reset the statement back to its original value.

Default SQL Select

This text box provides the base SELECT statement. Note that if your browse contain fields from more than a single table, it will automatically provide for a JOIN.

Important Note:

If you've populated a field in the Data Columns tab for which the ADO templates can't resolve the proper syntax (for example, if you've populated a data variable in the list box control for your browse) and the resulting page doesn't display properly, examine the SQL statement here. Should you find a statement with a blank for the field name (look for an extra comma in the order that the suspect field appears in the data columns list, as in Select fieldname,fieldname,,fieldname...), you may edit the statement here or delete the suspect column from the list. Such a problem should be very rare.

Unique Key

As stated on the tab control, ADO/SQL requires a unique key to identify a record. Should you have more than one unique key defined in your table, press the ellipsis button to select an alternate key to use.

Default Behavior

The default behavior of the ADO Browse Box has a much different interface than the standard browse box. You will also notice that the filter capabilities are expanded in the ADO template. The following prompts are available:

Fields Tab:

The Fields tab control provides control of your browse box filtering and sorting features.

Range or Filter Columns

Press the update buttons to add (Insert), modify (Properties), or remove (Delete) a range or filter column. These are the data elements used to limit the records displayed in your Browse Box. The Range or Filter dialog provides the following prompts:

Column
Select a column name from the drop list provided to use as criteria for the browse filter or range. The columns displayed are those columns you have defined in the Data Columns/Hot Fields dialog.

Use Static Value?
Check this box to use the static value of the column selected. This is the value of the column when the browse is first initialized

Negate the Range or Filter(NOT)
Check this box to apply the NOT clause to the selected column. This will have the effect of "reversing" the filter expression (Example: *MyCol = MyValue* will be generated as *WHERE NOT (MyCol = MyValue)*.

Range Limit Type
Specifies the type of range limit to apply. Choose one of the following from the drop-down list.

Single Value

Lets you limit the filter criteria to a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

In the case where you are using static values, you are responsible to enter the values with the necessary single quotes. For example, if you are using the **IN** filter criteria, and need to check several parameters enter *'Test', 'Test2','TestN'* in the **Single Value** line. For static values, you need to enter single quotes where needed.

Range of Values
Lets you specify upper and lower limits. Specify the variables containing the limits in the **Low Limit Value** and **High Limit Value** boxes.

Less Than
Lets you limit the records read to all records less than a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

Great Than

Lets you limit the records read to all records greater than a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

IN

Lets you limit the records read to all records that match the contents of a single value.

Begins with

Lets you limit the records read to all records that begin with the contents of a single value.

Ends with

Lets you limit the records read to all records that end with the contents of a single value.

Contains

Lets you limit the records read to all records that begin with the contents of a single value.

Sort Columns

Press the update buttons to add (Insert), modify (Properties), or remove (Delete) the browse sort column(s). These are the data elements used to sort the records displayed in your Browse Box. Clicking on the column header activate the column sort.

Add Primary Key Fields

Press this button to automatically include all fields defined in the primary key as sort columns.

Add a Key Field

Press this button to select any key from the primary table, and apply it to the sort columns.

The **Sort Columns** dialog also provides the following prompts:

Column

Select a column name from the drop list provided to use as criteria for the browse filter or range. The columns displayed are those columns you have defined in the Data Columns/Hot Fields dialog.

Direction

Choose ASC from the drop list to designate an ascending sort, or DESC to specify a descending sort.

SQL Tab Prompts

The SQL tab control displays SQL statements that are generated, based on the settings displayed on the Fields tab.

Essentially, the settings of the Range or Filter are used to generate the SQL WHERE clause, and the Sort Columns setting are used to generate the ORDER BY clause.

Regenerate WHERE SQL

Press this button to regenerate the template SQL WHERE statement that is constructed based on your default settings.

Override WHERE SQL

Check this box to override the default template generated WHERE clause.

Enter a WHERE clause to filter this list

The text in this box provides the Where clause, which is concatenated to the Select statement.

Regenerate ORDER BY SQL

Press this button to regenerate the template SQL ORDER BY statement that is constructed based on your default settings.

Override ORDER BY SQL

Check this box to override the default template generated ORDER BY clause.

Enter the ORDER BY clause

The text in this box provides the Order clause, which is concatenated to the other parts of the statement.

Test SQL Query

Press this button to test the SQL statement. If a statement clause is not overridden, the statement will be regenerated based on the template's current settings.

Conditional Behavior

The Conditional Behavior tab duplicates the prompts and resultant functions found on the Default Behavior tab, with the addition of a **Condition** prompt:

Condition

Enter any valid Clarion expression.

Colors

The Colors dialog of the ADO Browse template presents the exact functionality as the standard browse box.

Browse Box Colors

Icons

The Icons dialog of the ADO Browse template presents the exact functionality as the standard browse box.

Browse Box Icons

Styles

The Styles dialog of the ADO Browse template presents the exact functionality as the standard browse box.

Browse Box Styles

Tooltips

The Tooltips dialog of the ADO Browse template presents the exact functionality as the standard browse box.

Browse Box Tooltips

Totaling

The Totaling dialog of the ADO Browse template presents the exact functionality as the standard browse box.

Browse Box Tooltips

Classes

Use the Classes tab to override the global settings for the Class. See Classes Tab.

ADO Browse Box Select Button

The ADO BrowseSelectButton template provides Select button to choose a record from An ADO browse (list) box.

The generated source code gets the currently selected record from the list (makes the selected record the current one in the browsed file's record buffer), and closes down the procedure. For the end user, pressing the Select button is equivalent to double-clicking an item in the list.

The BrowseSelectButton template provides the following prompts:

Hide the Select button when not applicable

Check this box to hide the Select button when the procedure is not called for selection purposes (GlobalRequest <> SelectRecord).

Allow Select via Popup

Check this box if you would like to include the Select option in the Browse Box popup menu.

ADO Browse Process Button

The ADO BrowseProcessButton template provides a button to call an ADO Process or Report Procedure. This control acts on the records in a browse box. When pressed, the button retrieves the recordset and invokes the respective database action for that record.

The ADO BrowseProcessButton template provides the following prompts:

Process Procedure

Type a procedure name or select a procedure name from the drop-down list. If you type a new procedure name, the Application Generator adds the new procedure to the Application Tree. The procedure that you call will expect an ADO Process Control template to exist.

Procedure Parameters

Allows you to specify parameter names (an optional list of variables separated by commas) for your process procedure, which you can pass to it from the calling browse procedure. You must specify the functionality for the parameters in embedded source code.

Example:

```
(LOC:HideID,GLO:AccessLevel)
```

ADO Browse Update Button control template

The ADO BrowseUpdateButtons template provides three buttons for managing ADO I/O for the ADO BrowseBox: Insert, Change, and Delete. These three button controls act on the records in a browse box. When pressed, the button retrieves the selected record and invokes the respective database action for that record.

The ADO BrowseUpdateButtons template provides the following prompts:

Update Procedure

Type a procedure name or select a procedure name from the drop-down list. If you type a new procedure name, the Application Generator adds the new procedure to the Application Tree.

Procedure Parameters

Allows you to specify parameter names (an optional list of variables separated by commas) for your update procedure, which you can pass to it from the calling browse procedure. You must specify the functionality for the parameters in embedded source code.

Example: (LOC:HideID,GLO:AccessLevel)

Set UNIQUETABLE

This checkbox will control the generation of code to set the value of UNIQUETABLE in the recordset for updating. If checkbox is set, this will enable the UNIQUEVALUE Value prompt.

UNIQUETABLE Value

Press the ellipsis button to select the UNIQUETABLE (default value is the Primary file).

Generate a Resync statement

Check this box to force the ADO templates to generate a RESYNC statement. Some back ends (e.g., Oracle) do not need this statement specified. The statement generated will correspond to the one generated for the ADO browse box with an additional WHERE clause in which the Columns that compose the primary key are included.

Example: `SELECT * from Table where SysID = ?`

Call Resync after update

Check this box if the ADO recordset method RESYNC is called after an update. If set, the prompts **AffectRecords Parameter** and **Resync Values** parameter are enabled.

AffectRecords Parameter

This prompt defines the value that must be set for the **AffectRecords** parameter of the RESYNC command. Default value is *adAffectAll*, but some back ends may only work with *adAffectCurrent* (e.g., Pervasive).

Resync Values parameter

This prompt defines the value that should be set for the **ResyncValues** parameter of the RESYNC command. Default value is *adResyncAllValues*.

ADO Errors Object List Control Template

The ADO Error Object List control template displays a list of any ADO errors that are encountered during the login and connection attempt. The list box contains two elements:

Error Number – an error code returned from the ADO Connection layer

Error Description – a detailed description of the type of error returned.

This control needs to be populated on a procedure that is named by the *ADO Error Procedure* template prompt found in the ADO Login Extension or ADO Global Extension.

ADO Process Pause Button

This control template adds a button to allow pausing and restarting of an ADO Process or Report procedure.

Pause Text

The text to display on the button when the process is running. This text alerts the user that the process can be paused by pressing the button. The default is *Pause.*

Restart Text

The text to display on the button when the process is paused and multiple starts are allowed. This text alerts the user that pressing the button can restart the process. The default is *Restart.*

Start Paused

The state of the control when the procedure starts. If check, the process is paused until the user presses the button.

Start Text

The text to display on the button when the procedure opens. This text alerts the user that the process can be strated by pressing the button. The default is *Go.*

Allow Multiple Starts

Check this button to allow the process to start more than once.

When Pressed

The standard set of prompts for buttons. Normally, when using a Control template, these prompts are not used.

ADO Save Button Control Template

The ADO Save Button Control Template is used to update a series of data elements through an ADO connection. There are several option that are unique and different from the standard Save Button used in ISAM and SQL database updates.

In the Save Button properties (found on the button's Actions tab control) press the **ADO/SQL Behavior** button to access the ADO/SQL Behavior dialog. The following prompts are available.

General Tab

Connection Group:

Connection Object

Choose a Connection object name from the drop list provided. This list should contain the connection object you created in the ADO Global support template.

Use a New Connection

If you do not wish to use any of the Global Connections that are available, check this box to create a new connection specifically for this browse box.

New Connection

Press this button to call the Connection Builder. You also have the option to create (derive) a new Connection object to use instead of the global object; check the **Generate a COMIniter object** check box to enable this feature.

Command Object

The ADO connection layer contains a default command object that is responsible for handling the appropriate data update behavior. If you need to select an alternative object name, enter the value here.

ADO Error Handling

Specify an Error handler Procedure

Check this box if you would like to call an Error Handling procedure if any connection problem is noticed during the Save process. Select a procedure name, or enter a new one in the drop list provided.

Table Mapper – BASETABLENAME is not supported

Check this box if your ADO data source does not support BASETABLENAME capability.

In ADO, when you have created a SELECT statement through your template design that contains the same column name, but coming from a different table, the fields' collection will have field objects with the same name.

For example,

```
SELECT Customer.SysID, order.SysId etc…
```

The Fields collection above will have 2 Field objects with the name *SysID*. In order to make sure that the values from those 2 fields will go into the right application variable, we need to know which table the field belongs to. There is a property in the Field properties collection called ***BASETABLENAME*** that provide this value.

Columns Tab

The Columns tab control prompts are used to specify additional columns (fields) that need to be updated by the ADO Form, but are not populated on the window.

By default, all data elements populated on the window that are part of the ADO file will be automatically updated as needed.

Get records affected?

Check this box if you want to retrieve the number of records affected after an update command is issued by the ADO command object. This number will be the same number you would see if you issued an SQL update command and the backend responded with "*x* records were affected by the command.

In most cases, the records affected variable will return a 1 if the update to the ADO table was successful. A zero (0) will be returned if the update to the ADO table was not successful.

Records affected var

Select a variable name to store and process the Records Affected count.

Additional Columns to add in SQL statement

In addition to the fields (columns) populated on the ADO Form procedure window, you can add additional columns to update here. For example, you may need to add data to selected columns based on data entered on the form, but need to do it in source and hide it from the user.

SQL Tab

Override Generated SELECT

Check this box to bypass the auto generation of the SELECT statement by the templates. Your custom statement will be substituted in its place. This box also enables the **Regenerate SQL** button, should you wish to reset the statement back to its original value.

Regenerate SELECT

Press this button to reset the original template-constructed SQL statements. This is useful should you need to start from scratch again before customizing your statements. Only enabled when the next prompt is active (checked).

Classes Tab

The Classes Tab contains information regarding the names of classes used by the ADO form templates. For more information, see Classes Tab.

Help on the other Save button prompts are identical to the standard Save Button control template, and can be referenced there.

ADOLoginControls Control Template

The ADOLoginControls Control Template is used with the ADO Login Procedure to provide default login controls on the window. This includes:

UserID

Password

Server

Blank Password check box option

Login and Cancel Buttons

There are no template prompts associated with this control template.

ASCII Print Button control template

This Control template adds a button named Print, and the underlying code necessary for printing an ASCII (text) file. Use this control template together with the ASCII View control template .

Edit the **Actions** only if you wish to add another, separate action to take place *after* printing. All the code necessary for managing the print job itself is handled automatically.

The **Actions** tab contains the following:

When Pressed	The standard set of prompts for buttons. Normally, when using a Control template, these prompts are not used.

ASCII Search Button control template

This Control template adds two buttons named Find and Find Next, and the underlying code necessary for a modal search dialog, allowing the end user to find text in an ASCII (text) file. Use this control template together with the ASCII View control template .

Edit the **Actions** only if you wish to add another, separate action to take place *after* the search. All the code necessary for managing the search itself is handled automatically.

The **Actions** tab contains the following:

When Pressed	The standard set of prompts for buttons. Normally, when using a Control template, these prompts are not used.

ASCII View control template

The AsciiViewControl template adds a LIST control in which you can display read-only, the contents of a file--including variable length files. It is typically used to display an ASCII text file. The AsciiViewControl template optionally provides search and print capability for the displayed file.

The template lets you select the file to view at design time, or leaves the selection to the end user at runtime if you prefer. Finally, the template optionally allows the LIST control to alternate its display between the selected file and some other data that you specify.

The AsciiViewControl template provides embed points for its LIST control. It also provides the following prompts on the **List Properties** dialog **Actions** tab, the **Procedure Properties** dialog, or the **Extension and Control Templates** dialog:

General Options

Initialize Viewer

Determines when the procedure initializes the Viewer object. Initialization includes selecting the file to view, opening it, and reading it.

On Open Window
Initializes the Viewer when the window opens so that the Viewer's LIST is full upon initial display.

On Field Selection
Delays initializing the Viewer until the end user selects the Viewer's LIST control.

Manually
Does not initialize the Viewer. You must embed a call to the *Viewer#:Initialize* ROUTINE to initialize the Viewer.

File to Browse

Specifies the path and name of the file to view, or a variable containing the path and name of the file to view. The variable must be preceded by an exclamation point (!).

If no path is specified, the procedure looks for the file in the current directory.

If omitted (left blank), the Viewer object prompts the end user to select a file.

Reassign FROM attribute after Kill

Check this box to reset the Viewer LIST's FROM attribute after the Viewer shuts down. See *FROM* in the *Language Reference*. This lets you use a single LIST control to display both the **File to Browse** and other items as well.

Value or queue to assign

Type the label of the QUEUE (or the string constant) to assign to the Viewer LIST's FROM attribute.

Allow popup menu searching

Check this box to provide a (RIGHT-CLICK) popup menu choice to search the file.

Allow popup menu printing

Check this box to provide a (RIGHT-CLICK) popup menu choice to print some or all of the records in the file.

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

ASCII View in List box

The AsciiViewInListBox template allows a LIST control to alternate its display between a selected file and some other data that you specify.

The AsciiViewInListBox template provides the same functionality and the same prompts as the AsciiViewControl template. See *AsciiViewControl* for more information. The AsciiViewInListBox template provides one additional prompt. Because it is an Extension template and does not place its own control, the AsciiViewInListBox template prompts you for the LIST control to use to display text:

General Tab

List box field to use

Select the LIST control that alternates its display.

Initialize Viewer

Determines when the procedure initializes the Viewer object. Initialization includes selecting the file to view, opening it, and reading it.

On Open Window
Initializes the Viewer when the window opens so that the Viewer's LIST is full upon initial display.

On Field Selection
Delays initializing the Viewer until the end user selects the Viewer's LIST control.

Manually
Does not initialize the Viewer. You must embed a call to the *Viewer#:Initialize* ROUTINE to initialize the Viewer.

File to Browse

Specifies the path and name of the file to view, or a variable containing the path and name of the file to view. The variable must be preceded by an exclamation point (!).

If no path is specified, the procedure looks for the file in the current directory.

If omitted (left blank), the Viewer object prompts the end user to select a file.

Allow popup menu searching

Check this box to provide a (RIGHT-CLICK) popup menu choice to search the file.

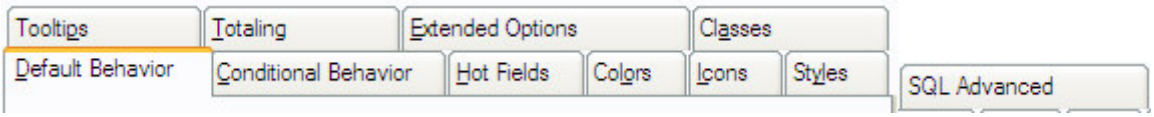
Allow popup menu printing

Check this box to provide a (RIGHT-CLICK) popup menu choice to print some or all of the records in the file.

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

Browse Box control template



Click on a tab to see its help

The BrowseBox Control template places a "page-loaded" or a "file-loaded" LIST control in a window and generates code to fill the list with data, and to scroll, search, sort, and select the listed items. It generates code to select or filter the data, total the data, update the data directly (edit-in-place), or call a separate procedure to update the data. It also generates code to conditionally set the colors and icons associated with each row and column in the LIST. The standard BrowseBox behavior is defined by the ABC Library's BrowseClass. See *BrowseClass* in the *ABC Library Reference* for more information.

You can use the BrowseBox Control template to manage a page-loaded *drop*-list simply by setting the DROP attribute to a value greater than zero (0) .

The LIST control's popup menu takes you to the **List Box Formatter** where you can choose which fields or variables populate the list. You can also define how they appear in the list box (including enabling colorization and Icon display). The **Actions** tab on the List Properties dialog provides the prompts which let you define the browse box's functionality, including any record filters, range limits, totaling, scroll bar behavior, and locator behavior.

Placing a BrowseBox on your window

You can place the *BrowseBox* Control template in a window by clicking on the template control tool , then selecting **BrowseBox - File Browsing List Box** in the **Select Control template** dialog. After you select the BrowseBox template, the Application Generator automatically opens the **List Box Formatter** so you can choose the files, fields and variables to display in the list, and you can design the appearance of the list and its fields.

Populating and Formatting the List Fields

The **Populate** button lets you add a field or variable to the list box, one field or variable at a time. The **Select Column** dialog presents the file schematic. Within the schematic, the BrowseBox control appears, with a <To Do> beneath it. To add a field from a data file defined in the dictionary:

Select the <*To Do*> item.

Press the **Insert** button

Select the file from the **Insert File** dialog.

If you want to use a Key, press the **Key** button to select the key from the **Key Access** dialog. If you do not select a Key, the list is displayed in record order, which also disables the ability to set Range Limits.

Select a field from the **Fields** list, which appears in the right side of the **Select Field** dialog. After you select the file, key and field (or variable) the **List Field Properties** dialog appears. This lets you precisely define the fields appearance within the list.

Actions

The **Actions** tab of the **List Properties** dialog (right-click the control and choose **Actions**) displays the template prompts which let you specify numerous template options, as well as add custom embedded source code for standard list box events, such as moving the selection bar. The dialog contains the following options:

Default Behavior

Table Schematic Description

Enter a descriptive string that will be displayed in the Table Schematic window for this particular control. This allows you to distinguish one control from another when there are multiple controls populated in a single window.

Quick-Scan Records

Specifies buffered access behavior for file systems that use multi-record buffers (primarily ASCII, BASIC, and DOS). See *Database Drivers* for more information. These file drivers read a buffer at a time, allowing for fast access. In a multi-user environment these buffers are not 100% trustworthy, because another user may change a record between accesses. As a safeguard, the driver refills the buffers before each record access.

Quick scanning is the normal way to read records for browsing. However, rereading the buffer may provide slightly improved data integrity in some multi-user circumstances at the cost of substantially slower processing.

Loading Method

Select the method used to read the BrowseBox data from the drop-down list.

Page
Page-loading provides near-instantaneous displays for unfiltered data, even for very large datasets. Page-loading uses less memory, because only a few records are held in memory at a time. On the other hand, page-loading can cause erratic vertical scroll bar behavior as well as additional network traffic for each scroll or search action.

File
File-loading provides smooth, accurate vertical scroll bar behavior, plus no additional network traffic when scrolling and searching. File-loading is also quite SQL friendly. On the other hand, file-loading can result in substantial delays and heavy memory usage when reading large datasets.

Accept browse control from Toolbar
Check this box to accept navigation events and other browse control events generated by the *FrameBrowseControl* control template on the APPLICATION's toolbar. See *FrameBrowseControl* for more information on these toolbar buttons and their operation. Clear this box to disable the *FrameBrowseControl* toolbar buttons for this procedure and use local navigation controls only. See also SetToolbarTarget.

LIST Line Height
Enter a number in dialog units (unless PROP:Pixels is active) to set the line height of each row generated in the browse box. This option sets the PROP:LineHeight property. You can optionally press the "E" button to call the Expression Editor, which is used to help you construct syntactically correct expressions or variables, used to calculate the desired LIST Line Height.

Locator
A locator lets the user search for specific records in the list box without manually scrolling through the entire list. **Locator** is only available when browsing a file in Key Order (specify a KEY in the Data / Tables Pad). The search field must be the first free key element, that is, the first component field of the browse key that is not range limited to a single value.

For multi-key browses (the Wizards create them), you may have multiple locators. Use the **Conditional Behavior** tab to set additional locators for the additional sorts. Choose from the following locator types in the drop-down list:

None	Specifies no locator.
Step	Specifies a single-character locator with no locator control required. When the BrowseBox has focus and the user types a character, the list box advances to the first occurrence of the key field beginning with that character (or the next higher character if no keys match the locator character). Retyping the same character advances the list to the next occurrence of the key field beginning with that character.

Use a step locator when the first free key element is a STRING, CSTRING, or PSTRING and you want the search to take place immediately upon the user's keystroke. Step locators are not appropriate for numeric keys. If there is no browse key, the Application Generator converts to no locator. Step locators are case-sensitive only if the key defined as the browse sort is defined as case-sensitive.

Entry

Specifies a multi-character locator that activates when the locator control is *accepted* (not upon each keystroke). The locator control may be an ENTRY, COMBO, or SPIN. Use an Entry locator when you want to search on numeric or alphanumeric keys, and delay the search until the user accepts the locator control (presses ENTER or TAB). This delayed search reduces network traffic and provides a smoother search in a client-server environment.

The locator control should come *after* the LIST control in the **Set Control Order** dialog.

By default, the locator control is the control whose USE attribute is the first free key element of the browse key. A free component is one that is *not* range limited to a single value. If there is no such control, the Application Generator converts to a Step locator. If there is no browse key, the Application Generator converts to no locator.

When the end user places one or more characters in the locator control, then *accepts* the control by pressing TAB, pressing a locator button, or selecting another control on the screen, the list box advances to the nearest matching record.

Incremental

Specifies a multi-character locator, with no locator control required (but strongly recommended). Use an Incremental locator when you want to search on numeric or alphanumeric keys and you want the search to take place immediately upon the user's keystroke.

The locator control may be a STRING, ENTRY, COMBO, or SPIN, however, any control other than a STRING causes the Incremental locator to behave like an Entry locator--the search is delayed until the control is accepted.

With a STRING control, when the list has focus, characters are automatically placed in the locator string for each keystroke, and the list box *immediately* advances to the nearest matching record. The backspace key removes characters from the locator string.

We strongly recommend using a STRING control as the Incremental locator control so the search occurs *immediately* with each keystroke, and so the user can *see* the key value for which the BrowseBox is searching.

By default, the locator control is the control whose USE attribute is the first free key element of the browse key. A free component is

one that is *not* range limited to a single value. If there is no such control, the Application Generator converts to a Step locator. If there is no browse key, the Application Generator converts to no locator.

Filtered

Specifies a multi-character locator, with no locator control required (but strongly recommended). Use a Filter Locator when you want to search on alphanumeric keys and you want to *minimize network traffic*.

This locator is like an Incremental Locator with a record filter. It specifies a *range* of values for which to search and returns a *limited* result set--only those records that fall within the specified range. Each additional (incremental) search character supplied results in a smaller, more refined result set. For example, a search value of 'A' returns all records from 'AA' to 'AZ'; a search value of 'AB' returns all records from 'ABA' to 'ABZ'.

The Filtered Locator determines the boundaries for the search based on the user specified search value. The implementation of the boundaries depends on the database driver--for SQL databases, the Filter Locator uses a LIKE; for ISAM databases it supplies upper and lower bounds.

The locator returns *only* the records that match the search value, providing, in effect, a dynamic range limit or filter for the browse.

The Filter Locator performs very well on SQL databases and on high order key component fields; however, performance may suffer if applied to non-key fields or low order key fields of non-SQL databases.

Override default locator control
The *default* locator control is the control whose USE attribute is the first free key element of the browse key. To override this default and specify a different locator control, check this box. This option is provided in case you have multiple controls with the same free key element as their USE attributes--that is, when you have both ascending and descending keys on the same field.

Select one of the controls to use as the locator control from the **New Locator Control** list.

Find Method
Only available when the Filtered locator is selected. The **Find Anywhere** checkbox determines whether the FilterLocator applies the search value to the entire field (field *contains* search value) or only to the leftmost field positions (field *begins with* search value). If checked, it applies the "contains" test. If unchecked, it applies the "begins with" test. See FloatRight for more information.

Locator Class
Press this button to override the global Locator Manager setting. See Classes Tab.

Record Filter

Type a valid Clarion expression to limit the contents of the browse list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter. Filters are generally much slower than Range Limits.

You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.

Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the Record Filter prompt.

Range Limit Field
In conjunction with the **Range Limit Type**, specifies a record or group of records for inclusion in the list. Choose a field by pressing the ellipsis (...) button. The range limit is key-dependent. Range Limits are generally much faster than filters.

Range Limit Type
Specifies the type of range limit to apply. Choose one of the following from the drop-down list.

Current Value
Limits the key to the current value of the **Range Limit Field**.

Single Value
Lets you limit the key to a single value. Specify the variable containing that value in the **Range Limit Value** box which appears.

Range of Values
Lets you specify upper and lower limits. Specify the variables containing the limits in the **Low Limit** and **High Limit** boxes.

File Relationship
Lets you choose a range limiting file from a 1:MANY relationship. This limits the list to display only those child records matching the current record in the parent file. For example, if your list was a list of Orders, you could limit the display to only those orders for the current Customer (in the Customer file).

See Also: Using Range Limits and Filters

Additional Sort Fields

Specify fields to sort on in *addition* to any Key specified in the **Data / Tables Pad** by typing an ORDER expression list (a comma delimited list of field names). See ORDER in the *Language Reference* for more information.

Reset Fields button
Press this button to add Reset Fields. If the value of any Reset Field changes, the procedure refreshes the BrowseBox list. Many BrowseBox events automatically refresh the list; however, if some want another control (such as a Radio button or an entry field) or process to refresh the list, use a Reset Field.

Scroll Bar Behavior button
Pressing this button displays a dialog where you can define the way a scroll bar works.

Choose from Fixed Thumb or Movable Thumb.

For file loaded lists, you automatically get Standard Windows standard (movable thumb) scroll bar behavior. However, since this is not possible for page loaded lists, these options let you choose the behavior that best suits your application.

Fixed Thumb
The thumb (square 3D box in the middle of the scroll bar) remains in the center of the scroll bar. CLICK above the thumb to scroll up one "page." CLICK below the thumb to scroll down one "page." DRAG the thumb to the top or bottom of the scroll bar to scroll the top or bottom of the file.

Choose Fixed Thumb when browsing large SQL tables to get best performance.

Movable Thumb
CLICK and DRAG the thumb to scroll a proportional distance in the list. The thumb remains where you drag it, and its

position on the scroll bar indicates the relative position within the browse list.

CLICK above the thumb to scroll up one "page." CLICK below the thumb to scroll down one "page".

When you choose **Movable Thumb**, you can also set the **Key Distribution** to further define how the BrowseBox evaluates the thumb's relative position within the browse list.

Key Distribution
Specifies the distribution of the points of the scroll bar. Choose one of the two predefined distributions (Alpha or Last Names), or Custom, or Run-time from the drop-down list.

Alpha
Defines 100 evenly distributed points alphabetically.

Last Names
Defines 100 points distributed as last names are commonly found in the United States. If the access key is numeric, you should use a custom or run-time distribution.

Custom
Lets you define your own points.

Run-time
Reads the first and last record and computes the values for 100 evenly distributed points in between.

Custom Key Distribution
Lets you specify the break points for distribution along the scroll bar (useful when you have data with a skewed distribution). Insert the values for each point in the list. String constants should be in single quotes (' ').

Run-time Distribution Parameters
Lets you specify the type of characters considered when determining the distribution points. This is only appropriate when the Free Key Element is a STRING or CSTRING. Check the boxes for the types of characters you wish to include for consideration. Choose from **Use alpha characters** (Aa-Zz), **Use numeric characters** (0-9), and **Use other keyboard characters.**

Step Class Press this button to override the global Step Manager setting. See Classes Tab.

Conditional Behavior

This tab contains a list box that lets you define BrowseBox behavior based on conditions or expressions. Add expressions to the list by pressing the **Insert** button. This displays a dialog where you define the expression and the associated behavior when that expression evaluates to true (nonzero or non-blank).

At run-time the expressions are evaluated, and the behavior for the first true condition in the list is used.

In this dialog you can specify:

Condition	Any valid Clarion expression. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the Condition prompt.
Key to Use	Optionally, the Key to use to sort the BrowseBox data when the expression is true.

The remaining fields and buttons are the same as the **Default Behavior** tab.

Hot Fields

When you select the Hot Fields tab, you can specify fields not populated in the list to add to the QUEUE. When scrolling through the file, the generated source code reads the data for these fields from the QUEUE, rather than from the disk. This speeds up list box updates.

Specifying "Hot" fields also lets you place controls outside the BrowseBox that are updated whenever a different record is selected in the list box. Elements of the Primary Key and the current key are always included in the QUEUE, so they do not need to be inserted in the Hot Field list.

This dialog also lets you BIND a field. You must BIND any file field, variable, or EQUATE that is used in a filter expression.

If the field you are BINDing does not need to be included in the default Browse VIEW structure, check the **Not in View** checkbox.

Colors

This tab is only available if you check the **Colors** box in the *List Box Formatter*. It displays a list of the BrowseBox columns that may be colored.

To specify the default colors and any conditional colors, highlight the column's field name, then press the **Properties** button. This opens the **Customize Colors** dialog.

Use same color for all columns

Check this box if the color assignments designated here will be for all columns.

Customize Colors

This dialog is available if the **Use same color for all columns** check box is cleared, and lets you specify the default and conditional Foreground and Background colors for normal (unselected) and selected columns.

Create Greenbar Effect

Check this box to allow two color sets to be used in an alternating row format. If this box is cleared, you can use the **Conditional Color Assignments** dialog to specify any number of colored rows based on specified conditions.

Alternate columns

Only enabled when the **Create Greenbar Effect** check box is checked. Check this box to specify that the two color pattern will be used to each column display in addition to each row (producing a checkerboard pattern).

Conditional Color Assignments

Below the default colors section is the **Conditional Color Assignments** list. This list lets you set colors to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the **Insert** button.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Icons
This tab is only available if you check the **Icons** box in the List Box Formatter. It displays a list of the BrowseBox columns which can display icons.

To specify default icons and any conditional icons, highlight the column's field name then press the **Properties** button. This opens the **Customize BrowseBox Icons** dialog.

Customize BrowseBox Icons

This dialog lets you specify the default icon and conditional icons for the BrowseBox column.

Default Icon
The default icon to display. Type the icon (.ICO) filename.

You can also name a variable to use as the default icon, using the !*variable* format. The variable may be a string type that stores the icon filename, or check the **The expression is a number** check box if you are referencing a previously named icon assigned to the IconList property.

Conditional Icon Usage
Below the **Default Icon** section is the **Conditional Icon Usage** list. This list lets you set icons to apply when an

expression evaluates to true (nonzero or non-blank). To add an expression and its associated icon, press the **Insert** button.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Styles

This tab is only available if you check the **Style** box in the List Box Formatter. It displays a list of the BrowseBox columns that may have applied styles.

To specify the default styles and any conditional styles, highlight the column's field name, then press the Properties button. This opens the Customize BrowseBox Styles dialog. A default style may also be defined on the List Box Formatter Appearance tab.

Create GreenBar Effect

Check this box to create a GreenBar effect (alternating colors on each row) on your list box. You will be prompted to select two styles to use that represent the appearance of each alternating row

Alternate Columns

Check this box to apply an alternate style to every other column defined with the style attribute. You will be prompted to select two styles to use that represent the appearance of each alternating column.

Style Type

Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number

Based on the Style Type selected, select a style's description or number from the drop list control.

Default Style

This entry lets you specify the default style to be used for the column.

Conditional Styles

This list lets you define the styles to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the Insert button.

Condition
Provide a valid Clarion expression that when evaluates to true (nonzero or non-blank) will cause the Style to be applied.

Style Type
Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number
Based on the Style Type selected, select a style's description or number from the drop list control.

At run-time the expressions are evaluated, and the styles for the first true expression are used.

All Styles must be defined. For more information on defining styles see Listbox Styles.

Tooltips

This tab is only active if you check the **Tooltip** box in the List Box Formatter. It displays a list of the BrowseBox columns which may have applied tool tips. Press the **Properties** button to display the *Customize BrowseBox Tooltips* dialog.

Tooltip variable

You can specify a default tool tip string value in the List Box Formatter. This entry lets you override the default tooltip to use a value contained in a variable. Press the ellipsis button to select a variable that will contain the text of your column's tool tip.

Totaling

This tab contains a list box that lets you define total fields for a BrowseBox.

Always Calculate Totaling?
Check this box to always calculate browse totaling fields each time the browse box is refreshed. If this box is unchecked, you can apply a condition to the calculation.

Condition
Enter a variable or expression on this line. If the variable or expression evaluates to a non-zero value, the browse totaling loop will be executed, and all total values updated. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the Condition prompt.

Press the **Insert** button to add total fields. This opens the **Browse Totaling** dialog where you can define total fields for the BrowseBox.

Total Target Field
The variable to store the calculated total. This can be a local, module, or global variable. You may also use a file field; however, you must write the code to update the file.

Total Type
Choose **Count, Sum,** or **Average** from the drop-down list. **Count** tallies the number of records. **Sum** adds the values of the Field to Total. **Average** determines the arithmetic mean of the Field to Total.

Field to Total
The field to sum or average. This box is disabled when the Total Type is **Count**. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Field to Total** prompt.

Total Based On
Choose **Each Record Read** or **Specified Condition** from the drop-down list. This specifies whether to consider every record or only those that meet the Total Condition criteria.

Total Condition
The condition to meet when using a Total based on a specified condition. You can use any valid Clarion expression. You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Total Condition** prompt.

Extended Options

Do not include Primary key in view

Check this box to specify that primary key components are not projected into the view. This is useful when working with SQL tables, and allows valid GROUP BY SQL statements to be generated. See also SQL Advanced Tab

Disable Browse Popup menu

Check this box to disable the popup menu for this Browse Box.

Enable Sort Header (ABC Template Chain Only)

Check this box to allow your list box header to determine the sort order of your list box.

At runtime:

- Enable a sort on any column by CLICKing on the appropriate column header.

CLICK on the list header again to toggle between an ascending or descending sort.

- Press CTRL + CLICK on a list header column to add that column to the sort.
- Press SHIFT + CLICK on any list header column to remove all current sorts. Removing all column sorts restores the list box to the default sort.

This option is compatible with the List Format Manager and the Auto-Size option discussed below.

This option is not valid with MEMO, BLOB or fields not defined in the browse VIEW structure.

Customize BrowseBox Sort Header

Press this button to access a dialog that allows you to control the sort headers for each individual column. You can disable any or all columns, and specify custom sorts for local variables and other special program conditions. See **Customize BrowseBox Sort Header**

Disable Auto Size BrowseBox Columns?

If you have enabled the Auto Size Column feature in the Global Properties, check this box if you wish to disable this feature for this procedure's Browse Box.

Disable List Format Manager?

If you have enabled the List Format Manager in the Global Properties' App Settings, check this box to disable this feature

List Format Manager

If the List Format Manager is enabled in the Global Properties, and not disabled, press this button to access the List Format Manager dialog.

IPDRV Options

If you are using the IP Driver in your application, press this button to access the IPDRV Options dialog.

SQL Advanced Tab

When the BrowseBox Control template is using an SQL Accelerator driver, SQL Accelerator Drivers convert standard Clarion file I/O statements and function calls into optimized SQL statements, which they send to their backend SQL servers for processing. This means you can use the same Clarion code to access both SQL tables and other file systems such as TopSpeed files. It also means you can use Clarion template generated code with your SQL databases.

This tab control (which only appears when you are using an SQL table) allows you to extend the optimized SQL statements generated by the template.

The following options are available:

Query Elements

The Query Elements items allow you to assign special SQL clauses to your browse box result set. This is accomplished by selecting an existing element in the SQL table, and assigning it to a special function or expression. See Runtime SQL Properties for Views using SQL Drivers

View Field

Select an SQL column from the drop list provided.

Assignment
Enter a valid SQL statement to assign to the View Field.
Example: `'count(*)'`

The example above is equivalent to:
"SELECT count(*) FROM *tablename*"

Grouping

If you have enabled the **Do not include Primary key in view** option in the Browse controls' *Extended Options*, the following options are enabled.

Group
To add a GROUP BY clause to the template generated SQL statement, check this box and enter the appropriate SQL string in the **Grouping Definition** field. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Group** prompt.

Having
To add a HAVING clause to the template generated SQL statement, check this box and enter the appropriate SQL string in the **Having Definition** field. You must set a GROUP BY definition first to enable the HAVING clause. Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Having** prompt.

See GROUP BY and HAVING and VIEW support for aggregate functions for more information and examples.

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

BrowseFuzzyMatching control template

The BrowseFuzzyMatching control template adds a set of controls (a GROUP box containing an ENTRY control and two BUTTON controls, Search and Clear), that lets end users perform weighted searches and sorts of the result.

The query examines the columns in the BrowseBox queue, (ListBox fields and Hot Fields), and returns the data sorted in their order of relevance to the search criteria. At runtime, the BrowseBox may contain an extra column that shows a calculated match result. Prior to searching, the initial match result is 100%. No match to the search criteria gives a 0% match result.

Requirements

This template requires a BrowseBox control.

Populating the Control Template

Open the Window Designer for any procedure containing a BrowseBox control.

Select Populate ▸ Control Template.

The Select Control Template dialog displays.

Highlight *BrowseFuzzyMatching* and press the **Select** button.

Click on the window at the position where you want the Fuzzy Search controls to be placed. This populates an ENTRY control, two BUTTON controls, and a GROUP box surrounding them.

Modify the text and properties of these controls as desired. For example, you may want to change the text of the GROUP box to read "Search Personnel Records" instead of the default text.

RIGHT-CLICK on the GROUP box, then select **Actions** from the popup menu to set the Fuzzy Match options.

Template Prompts

The BrowseFuzzyMatching template provides the following prompts:

Display Match Results in Browse Listbox	Check this box to display the match result percentage in the Browse Listbox.
Display Match Results in Browse Listbox	Check this box to display the match result percentage in the Browse Listbox.
Display Where	Specifies the position of the match result column in the Browse Listbox. Choose from:
First Column	Populates the match result column as the first column in the Browse Listbox.
Last Column	Populates the match result column as the last column in the Browse Listbox .
Column Format	Specifies the display format of the match result column. The formatting information is set using the same syntax as the FORMAT() attribute for a LIST control.

Match Score Filter Value	Specifies the minimum match value to display in the Browse Listbox. Using a value of 1 will filter out any data that has no matches to the search data.
Reset Fuzzy Match Result Order on Control Event?	Check this box to reset the Browse Listbox results from another control and event on the window.
Control	Choose a control from the droplist that will control the reset of the query.
Event	Choose the event (*Selected* or *Accepted)*, which will force a reset on the query when it occurs on the selected control.
Fuzzy Group Resize Options	Allows the definition of the resizing techniques to use for the group of populated controls.
Resposition Strategy	Specify how the horizontal and vertical position is determined when the end user resizes the window. Valid reposition equates should be used. The reposition EQUATE's are defined in ABRESIZE.INC. The default is set to Resize:FixNearestX + Resize:FixNearestY.
Resize Strategy	Specify how the control's height and width is determined when the end user resizes the window. Valid resize equates should be used. The resize EQUATE's are defined in ABRESIZE.INC. The default is set to Resize:LockSize

Global Options

From the Application Tree, press the **Global** button.

Select the **General** tab if it is not already selected.

The following prompts are available from the Global Options dialog, **Enable Fuzzy Matching**, and Fuzzy Matching Options—**Ignore Case** and **Word Only**. See *Template Overview—Global ABC Template Settings—General Tab Options*.

BrowseGrid

The BrowseGrid template provides a group control that is used to define an area (within a BrowseBox) to display controls in a matrix format. The size of the BrowseGrid control and BrowseBox can be manipulated to give the end result of a matrix with one column or many. When this template is added, the BrowseBox control does not display at runtime.

This feature is particularly useful for Web applications, but can also extend the capabilities of desktop applications.

Requirements

This template requires a BrowseBox control template. The BrowseGrid GROUP box must be placed entirely within the BrowseBox. If the application is to be a Web application, the WebGridExtension must be added to the procedures extension set.

Populating the Control Template

Open the Window Designer for any procedure containing a BrowseBox control.

Select **Populate ▸Control Template.** The *Select Control Template* dialog displays.

Highlight *BrowseGrid* and press the **Select** button.

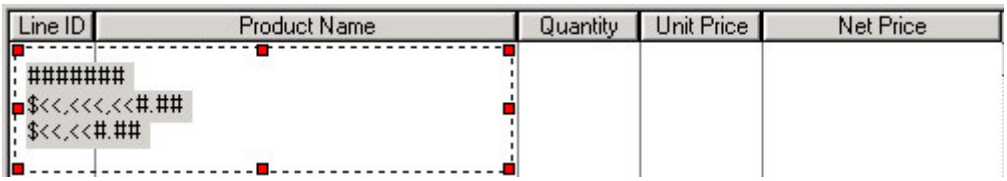
Click on the window at the position where you want the Browse Grid.

Note:

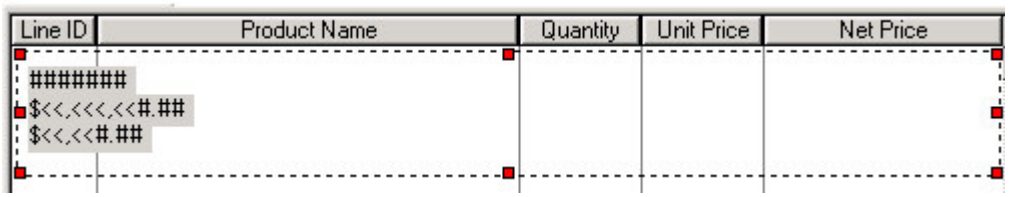
If you have a multi-tab Browse, you do not want to populate the control template inside the TAB structure. Populating it outside of the structure will enable it to display for all tabs.

Resize the *BrowseGrid* Group box, as desired. The size of the LIST and the size of the GROUP determine how many columns will appear at runtime. The smaller the GROUP width, the more columns will be displayed at runtime.

This layout will produce a list with two columns and multiple rows (GROUP box ½ the size of the LIST box):



This next layout will produce a grid with one column and multiple rows:



Populate other controls you want to display for each record. Keep in mind that these are for display only unless you write the code to update records.

Typically, you will populate STRING controls in a BrowseGrid GROUP. You may also want to populate a DynamicImage control.

Template Prompts

The BrowseGrid template provides the following prompts:

Expression for Group Title
Specifies a string, variable, or runtime expression using EVALUATE to use as the title for the group box that outlines each entry in the grid. To specify a variable here, precede the entry with an exclamation point (!). To specify a runtime expression, precede the entry with an equal sign (=).

Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Expression for Group Title** prompt.

Color of Selected Group
Specifies the color of the currently selected group in the BrowseGrid. Type in a valid color equate, hex color code, or choose from the color dialog by pressing the ellipsis button.

If a color is specified in a Web application, the specific color used at runtime is taken from the web skeleton set.

Update on control event
Check this box to allow the **Update Control** use variable to update when the **Update Control Event** occurs.

Update Control
Specifies the control that will initiate the update.

Update Control Event
Specifies the event that triggers the update action.

Make Group Selectable
Check this box to indicate the group defined by the BrowseGrid can be selected and generate events.

Forward Other Control Clicks
Check this box to force an EVENT:Accepted action to the **Forward Click To** control when a cell in the matrix is selected. A DOUBLE-CLICK activates the action in a Windows application.

Forward Other Control Clicks (To)
Specifies the control to process when a cell in the BrowseGrid matrix is selected.

BrowsePrintButton

The BrowsePrintButton template provides a **Print** button to call a procedure with the ProcessRecord request (see *Procedure Templates--Inter-Procedure Communication* for more information).

Print All Items

If you use the BrowsePrintButton to call a simple Report procedure, the report prints as usual, applying any design-time keys, sort orders, range-limits, and filters.

Print the Selected Item

If you use the BrowsePrintButton to call a Report procedure with the ExtendProgressWindow extension template (set to Single record), the report reacts to the ProcessRecord request and processes only the selected BrowseBox item. See *Other Templates--ExtendProgressWindow* for more information.

This option works by using a current-value limit on the report key. Therefore, if you have a non-unique key you can print multiple items--for example, all customers named Smith.

The BrowsePrintButton template provides the following prompts:

Procedure name

Type the name of a procedure to call or select a procedure from the drop-down list.

BrowsePublishButton (BrowseToHTML)

The BrowsePublishButton template provides a Publish button to generate the Hypertext Markup Language (HTML) to display records from a BrowseBox queue. In other words, use this template to publish your BrowseBox information to an Internet Web page!

Note:

The BrowsePublishButton template is only available in the Clarion template chain and cannot be used with the ABC Templates.

The resulting Web page displays a headline that you specify, plus the headers from your list box. The Web page formats the list box data using the picture tokens specified in the list box.

At runtime, the user may specify the filename for the generated HTML. Also, the user has the option to publish all the items in the BrowseBox queue or just the items currently displayed on the screen.

The BrowsePublishButton template provides the following prompts:

Use variable for HTML name

Check this box to specify the HTML file with a variable. This enables the Variable HTML filename field to name the variable, and disables the Default HTML Name field.

Default HTML Name

Specifies the default filename for the HTML code or the variable that contains the HTML filename. Press the ellipsis (...) button to select the file from the standard Open File dialog, or to select or define a data dictionary column or memory variable from the Select Field dialog.

If you don’t specify a full path, your procedure writes the file to the current directory. At runtime, the user may specify a different file and path name.

HTML Title

Specifies the title for your HTML document. The title appears in the Web browser’s caption when it displays the document.

Table Heading

Specifies headline that displays at the top of the Web page.

Background Graphic

Specifies a graphic image that displays "behind" the queue items. Note, most Web Browsers support graphic images, however, some older versions do not.

Use Grid Lines

Check this box to display the queue items within a rectangular grid. Note, most Web Browsers support grid lines, however, some older versions do not.

Grid Line Width

Specifies the thickness of the border defining the grid.

BrowseNoRecords Button control template

The BrowseNoRecords button control template provides a standard button that is only available (enabled) when there are records in the target Browse Box control.

If there are no records in the Browse Box control, the button is disabled or hidden.

A popular use for this type of control is a report or process that is called, using the record highlighted in the Browse Box control as a filter criteria, or parent record. Another use of this type of control could be to modify the contents of the Browse Box (marking records, moving them, etc.)

After the button's actions are performed, the browse box can optionally be refreshed.

The BrowseNoRecords button template provides the following prompts:

Select from popup

Check this box to add the button control to the Browse Box popup menu. The text of the button is used as the popup menu item text.

When no Records

Select **Disable** or **Hide** to designate how you would like the button to be displayed when there are no records in the Browse Box control.

Refresh Window After Action?

Check this box to force a window refresh after the button's actions are completed. In the ABC chain, this will generate "ThisWindow.Reset(True)". In the Clarion template chain, this will generate "ForceRefresh = TRUE;DO RefreshWindow".

BrowseQBEList Control Template

Click here to jump to the Browse QBE List template prompts.

The Browse QBEList Control Template is a special control template that allows you to test table queries easily and view the results. You can drag columns from a browse box into the query area, enter values, and quickly test your data set.

This template requires that a Browse Box be populated on the window. Although there are no special prompts associated with this template, there are multiple controls that are populated together and are described below:

Filtering Center List Box

This is an EIP (edit-in-place) list box that allows you to enter a column name, operator, Value or Expression, and an optional connecting operator to the next query.

You can also drag a column name from the browse box to this list box.

You can also drag specific value from a selected row of the browse. For example, if you select the Column *Name* and drag from a row where the value of that column is *"Bob"*, the QBE List will use that column/value to create *Name = 'Bob'*.

In the Value or Expression line, do not enter a single quote in your search string. Instead, if you wish to search for ‘FL’ as your search string, enter ‘’FL’’.

You can query fields with NULL values on SQL tables using the QBEList by entering the following in the Value line:

<NULL>

Updates to the browse box forces an automatic refresh of the active query.

Case sensitive search for string

Check this box to force case sensitive string searches for the current active query.

Reset Button

Clears the list box for a new query.

Save Query Button

Saves the query (to the non-volatile storage source specified in the application’s global settings).

Storage is automatic with ABC templates, but you must specify a storage source in the Global Properties of your application when using the Clarion template chain.

Save As Button

Saves the query to a new name that you specify from a query previously loaded.

Load Query Button

Loads a previous query from the non-volatile storage source.

Apply Button

Executes the current active query.

Query Center Operator Translation

The operator names that are used with the Query Center are stored externally in a translation file named *Cfilterlist.trn* found in \LIBSRC folder. What this means is that the default list of operators (EQUAL, CONTAINS, BETWEEN, etc.) can now be customized outside of the program code in this translation file. The file *cFilterList.TRN* defines a group with the following format:

```
DefaultOpe  GROUP
Number       USHORT(20)
             PSTRING('Equal')
             PSTRING('Equal')
             ULONG
             [...]
            END
```

The first PSTRING is what the user will see in the Query Center. The second PSTRING is the operator that the source code will use internally. The ULONG represents a bit mapped value that will indicate to the source code what data type will be valid with the operator (not implemented in this release). The first *Number* in the group is just telling the Query Center how many repetitions are present in the group. In our example above, there are currently 20 operators available in the translation file.

This translation file serves two purposes. It allows for variations in what the end users see for spoken language translations, and provides the ability to add "natural language" queries.

For more detailed information regarding the translation file, see the Query Center Translation File topic.

Browse QBE List Template Prompts

The following template prompts are provided:

Filter class

The base class used for the QBE List. The default is **cFilterList**.

Filter object name:

The object name used for this procedure. The default name is **FilterObj**.

Copy generated filter string to clipboard

Check this box to enable the filter string generated by the Filter Class to be copied to the Windows Clipboard. Pressing the Apply button initiates this feature.

Search on string is case sensitive

Check this box to allow the QBE List to default to case sensitive searches.

Case sensitivity search can be set at runtime

Check this box to allow the **Case sensitive search for string** check box to be populated at runtime that allows a user to enable or disable case sensitive string searches.

Browse Select button control template

The BrowseSelectButton template provides Select button to choose a record from a list box.

The generated source code gets the currently selected record from the list (makes the selected record the current one in the browsed file's record buffer), and closes down the procedure. For the end user, pressing the Select button is equivalent to double-clicking an item in the list.

The BrowseSelectButton template provides the following prompts:

Hide the Select button when not applicable

Check this box to hide the Select button when the procedure is not called for selection purposes (GlobalRequest <> SelectRecord).

Allow Select via Popup

Check this box to allow record selection with a RIGHT-CLICK popup menu. The template adds a popup menu item whose text matches the text on the Select button. The menu item is disabled when the Select button is disabled or hidden.

BrowseToolboxButton

The BrowseToolboxButton template provides a **Toolbox** button. Pressing the button starts a floating, dockable toolbox containing buttons that invoke the BrowseBox actions defined by the BrowseBox popup menu (Insert, Change, Delete, Select, Print, etc.).

The BrowseBox template automaically adds the Toolbox choice to its popup menu; therefore you can HIDE the **Toolbox** button but still provide access to the toolbox with the popup menu.

The BrowseToolboxButton template provides no configuration prompts.

The standard Toolbox behavior is defined by the ABC Library's PopupClass. See *PopupClass* in the *ABC Library Reference* for more information.

BrowseToolbarControl

The BrowseToolbarControl template places thirteen (13) standard command buttons on the window. When the user presses these buttons, the template generated code posts appropriate events (scroll up, scroll down, add, change, delete, help, etc.) to the various Browse controls. This template requires a BrowseBox control. See *Control Templates – FrameBrowseControl* for detailed information on each of the thirteen button controls.

Classes

The classes tab lets you control the class (and object) the template uses. You may accept the default Application Builder Class and it's object (reccomended) or you may specify your own or a third party class. Deriving your own class can give you very fine control over the procedure when the standard Application Builder Class is not precisely what you need.

See *Template Overview – Classes Tab Options – Local* for complete information on these options.

BrowseUpdateButtons control template

The BrowseUpdateButtons template provides three buttons for managing file I/O for a BrowseBox: Insert, Change, and Delete. These three button controls act on the records in a browse box. When pressed, the button retrieves the selected record and invokes the respective database action for that record.

The BrowseUpdateButtons template lets you specify an update procedure (recommended for files with two-way relationships) or edit-in-place updates (recommended for lookup files--files with one-way relationships).

(Jump directly to the Edit-in-Place topics)

The BrowseUpdateButtons template provides the following prompts:

Update Procedure

Type a procedure name or select a procedure name from the drop-down list. If you type a new procedure name, the Application Generator adds the new procedure to the Application Tree.

Procedure Parameters

Allows you to specify parameter names (an optional list of variables separated by commas) for your update procedure, which you can pass to it from the calling browse procedure. You must specify the functionality for the parameters in embedded source code.

Example: (LOC:HideID,GLO:AccessLevel)

Press the "E" button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the **Procedure Parameters** prompt.

Allow Edit via Popup

Check this box to provide right-click popup menus on the Browse list in addition to any command or toolbar buttons.

Edit-in-Place

Use Edit in place
Check this box to let the end user update the browsed file by typing directly into the BrowseBox list. This provides a very direct, intuitive spreadsheet style of update. You may configure the Edit in place behavior with the **Configure Edit in place** button.

The *Detailed* Interface of the Browse EIP manager now enables an expanded variety of control types to be used with Edit In Place without the need to embed hand code.

In order to invoke the new EIP manager, go to *Global Properties* and select the Classes Tab. Click on the **Browser** button, and click on the **EIP** tab control. A **Template Interface** prompt has been added to the dialog on the EIP tab. From the drop down associated with that entry, select *Detailed*. If your application had previously existing hand code, these embeds will be archived as *Orphaned*.

Configure Edit in place
Press this button to open the **Configure Edit in place** dialog. This dialog provides the following prompts:

General Tab Prompts

Save
The **Configure Edit in place** dialog offers the **Save** option for four different keyboard actions. These options determine whether changes to an edited record are saved or abandoned upon the following keyboard actions: TAB key at end of row, ENTER key, up or down arrow key, focus loss (changing focus to another control or window, typically with a mouse-click). Choose from:

Default	Save the record as defined in the BrowseClass.Ask method.
Always	Always save the record.
Never	Never save the record, abandon the changes.
Prompted	Ask the end user whether to save or cancel the changes.

Remain editing
The **Configure Edit in place** dialog offers the **Remain editing** option for three different keyboard actions. Check these boxes to continue editing upon the following keyboard actions: TAB key at end of row, ENTER key, up or down arrow key. Clear the boxes to stop editing.

Retain column
The **Configure Edit in place** dialog offers the **Retain column** option for the up and down arrow keys only. Check this box to continue editing within the same list box column in the new row. Clear to continue editing within the left most editable column in the new row.

Insertion Point
The **Configure Edit in place** dialog offers the **Insertion Point** option for initial new record placement in the list. The droplist choices— *before, after,* and *append*— indicate where the edit-in-place row will appear in the list when inserting a record. *Before* and *after* indicate placement in relation to the highlighted record, and *append* places the edit-in-place row at bottom of the list.

This does not change the sort order. After insertion, the list is resorted and the new record appears in the proper position within the sort sequence.

Action on Delete
The **Configure Edit in place** dialog offers the **Action on delete** option when a record deletion is attempted during edit-in-place. Select *Prompted* from the drop list if you wish the user to be prompted with a confirm delete message prior to deletion. Select *Never* if you do not want to allow record deletions from an edit-in-place. Finally, select *Always* if you want to allow users to delete records during edit-in-place, and the deletion is automatic (no confirm message).

Field Priming on Insert
Press this button to access the Field Priming dialog window. Field Priming lets you provide a default value for fields in a new record. This value supersedes any initial value specified in the data dictionary. You can select a field and set an initial value in the **Field Priming** dialog.

Column Specific Tab Prompts

The interface that follows for column specific EIP options will vary according to the Global EIP Configuration option you have set.

By default, the Global EIP Configuration option is set to *Original*. This option presents you initially with a blank list box. You must specifically **Insert** each column that will be used for edit-in-place. The ability to override the default class behavior is also available as you Insert each column.

The *Detailed* Global EIP Configuration automatically populates the Column Specific list box with *all* columns. Pressing the **Properties** button allows detailed EIP control that is discussed below.

If the Global EIP Configuration is set to *Original*, the *Column Specific* dialog will appear without a **General** and **Class** tab control. These tab controls only appear when a *Detailed* Global EIP Configuration is selected.

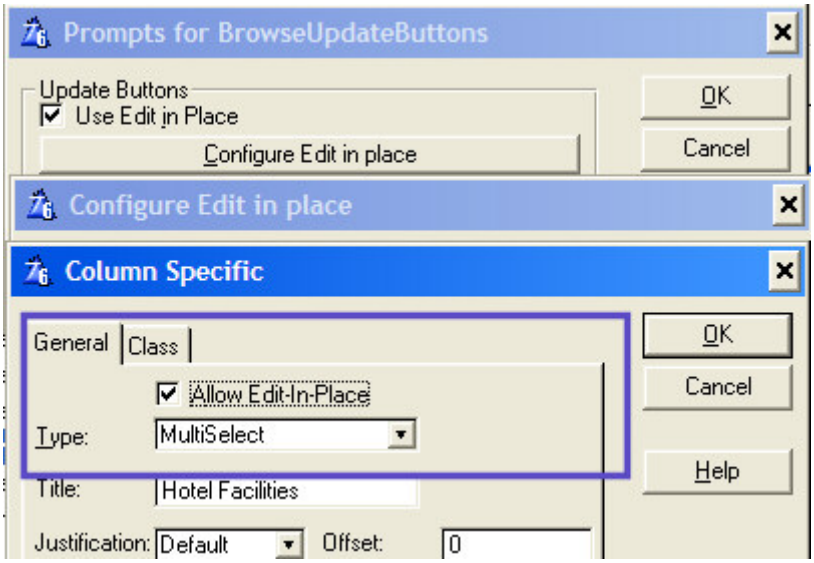
In *Original* EIP mode, press the ellipsis button to select a **Field** to add to the EIP controls. Check the **Allow Edit-In-Place** box to allow edit-in-place for the selected control. After selecting a field, the **Class Definition** group is then enabled.

In both modes, the *Column Specific* dialog lets you control the class (and object) the procedure uses to edit a specific Browsebox column. You may specify your own or a third party class.

By default, the BrowseUpdateButton template generates code to use the EditEntryClass in the ABC Library. The Global Edit in place (EIP) class must have the EditEntryClass specified as the base class. You can also use the other edit classes or derive your own.

The Application Generator must know about the CLASS you specify--see the *Template Overview--ABC Compliant Classes* section in the *Template Language Reference* for more information. See also Classes Tab.

Column Specific Dialog: General Sub Tab Prompts
This tab and its associated prompts are only available when the Global EIP Configuration is set to *Detailed.*



Allow Edit-In-Place
Check this box to allow edit-in-place for the selected control.

Type
Select from 13 types of window and dialog controls. Each control and dialog support has specific prompts that are applicable to that specific type.

The type control triggers the appropriate EIP class implementation, and options set the properties necessary for that class object. For more information, see the EditClass and other supporting classes in the ABC Library Reference.

It is also important to know that the *DropList* and *DropCombo* type controls used for EIP are manual controls (not control templates), and will need to have their list queues manually defined and populated to ensure their successful use.

The *MultiSelect* type includes a **MultiSelect Options** button where the following additional options are available:

Data Source
Choose from *Fixed* or *Queue*. This designates if the multiselect list box will be loaded from a hard coded (fixed) set of values, or from a queue that can be dynamically built and modified at runtime.

The Fixed option contains an option to designate an item as auto-selected. Click on the **Selected** check box to enable this option. This option is ignored if the **Auto Serialize Field** option is enabled.

Auto Serialize Field
Check this box if you would like the template to auto-populate the target EIP column with selected items in the *MultiSelect* dialog. If you do not enable this option, you will have to manually process the selected elements using the TakeCompletedProcess embed point. See the *ABC Library Reference* for more information.

Field Delimiter
Specify the delimiter to use between selected items. This prompt is required if you choose to auto serialize. Enter the delimiter *without* single quotes. A "space" delimiter cannot be used here.

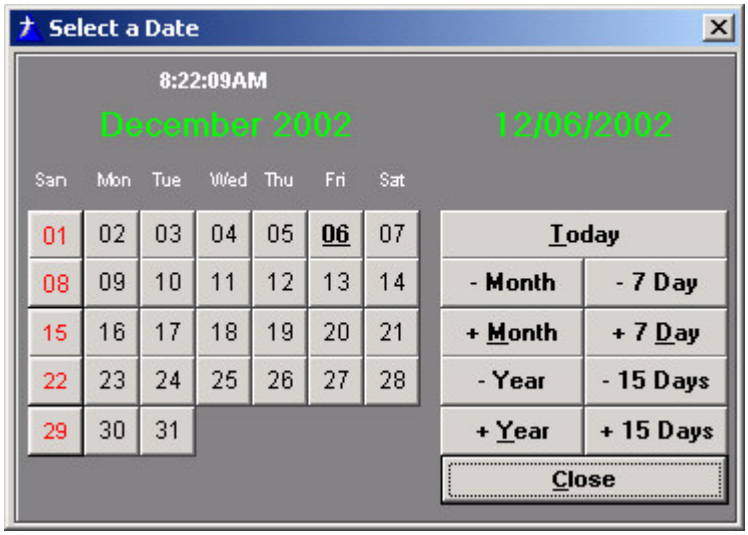
BrowseViewButton

The BrowseViewButton template provides a **View** button that calls the browse update procedure without update functionality (view only). This allows the end user an opportunity to look at a record and its associated child records via the update procedure while not permitting updates to the record.

This template adds a View button and the underlying code to call the update procedure. The code sets GlobalRequest to ViewRecord. The BrowseViewButton template requires an instance of a BrowseBox template, and the Browse Update Buttons. If you wish your browse to only have View Only capablilities, you can simply delete the update buttons.

The BrowseViewButton template provides no additional template prompts, but adds standard embed points for the View BUTTON.

Calendar Button Control Template



The Calendar Button Control Template is a button control that, when pressed, displays a full-feature calendar. A date value is returned to a targeted entry control. There are simple options to control the calendar title, extent of display features, and refresh window options.

Calendar Button - General Tab

The Calendar Button Control Template is a button control that, when pressed, displays a full-feature calendar. A date value is returned to a targeted control. There are simple options to control the calendar title, extent of display features, and refresh window options.

The CalendarButton template provides the following prompts:

Control

Select the associated ENTRY control for which to send the date value you selected on the calendar control.

Calendar Title

Enter a string value that you wish to use as the title of your calendar control.

Refresh Window

Check this box to force a refresh of the window controls when returning to the calling procedure.

See the Calendar Button - Classes Tab for additional information.

Calendar Button - Classes Tab

Many of the ABC Procedure, Control and Extension templates provide a Classes tab or dialog. These local Classes tabs let you control the classes (and objects) your procedure uses to accomplish the template's task—that is, they override the global class settings specified in the **Global Properties** dialog. You may accept the default Application Builder Class specified in the **Global Properties** dialog (recommended), or you may specify your own or a third party class to override the default setting. Deriving your own class can give you very fine control over the procedure when the standard Application Builder Class is not precisely what you need.

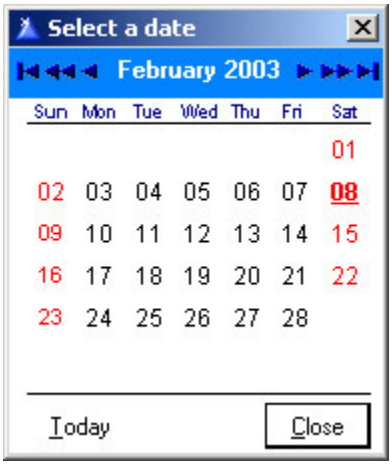
Click here to view the Global Calendar Configuration dialog.

Clarion provides you with three types of Calendar Classes that you may use with the CalendarButton control template, or, in hand coded projects if needed.

Select from the following classes:

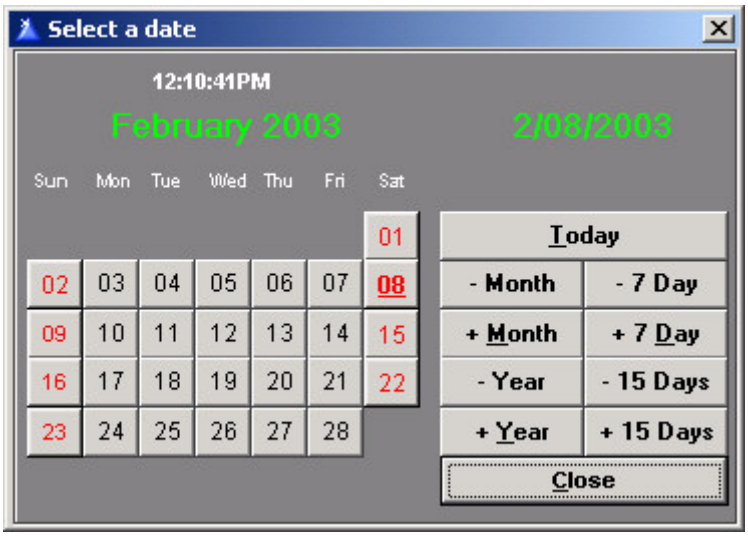
CalendarClass

The CalendarClass displays a simple and functional calendar that was designed to be compatible with the Windows XP look.



CalendarSmallClass

The CalendarSmallClass displays a calendar that is more compatible with other non-XP style environments:



CalendarBaseClass

The default source code for the CalendarBaseClass is located in the ABUTIL.CLW file (located in the LIBSRC directory). You must include the default window and corresponding source code in the respective DATA and CODE sections of the CalendarBaseClass derived ASK method. User interface structures and controls for popup calendar are located in ABUTILUI.INC.

Cancel Button control template

The CancelButton template adds a single button control marked **Cancel**. This button lets the user close a window and it provides a convenient place for the developer to add code to "undo" before closing down the procedure. The generated source code sets a "Request Cancelled" flag and closes down the window procedure.

The CancelButton template provides no configuration options.

You can insert the executable code you need to "clean up" at an embed point.

When Pressed The standard set of prompts for buttons Normally, when using a Control template, these prompts are not used.

Close Button control template

The CloseButton template adds a single button control marked **Close**. The generated source code sets a "RequestCancelled" flag and closes down the window procedure.

The CloseButton template provides no configuration options.

The **Action** tab provides the following prompts:

When Pressed The standard set of prompts for buttons Normally, when using a Control template, these prompts are not used.

DOS File Lookup control template

The DOSFileLookup template adds an ellipsis (...) button which opens the standard Windows file dialog.

You can specify the file masks, the default directory and filename, and the variable to receive the filename selected by the end user.

In addition, you may optionally allow the selection of multiple files and specify the code to process each selected file. The template generates a LOOP to process all the selected files.

Another new feature supported is the ability to assign the filename selected directly to an IMAGE control.

The DOSFileLookup template provides the following prompts:

General

File Dialog Header	Type the text for the caption of the Windows file dialog.
DOS Filename Variable	Press the ellipsis (...) button to choose a variable to receive the end user's choice from the **Select Column** dialog. You can also type the variable name directly into the entry box.
Default Directory	Specify the starting directory for the Windows file dialog. If blank, the file dialog opens to the working directory. If you want to use a variable for the pathname, then you must preface it with an exclamation point (Example: *!Myvariable*).
Default Filename	Specify the initial filename for the Windows file dialog. If blank, the file dialog opens with no initial filename.
Return to original directory when done	Check this box to reset the working directory to its value prior to the file lookup.
Long Filenames?	Check this box to enable the Windows file dialog to allow the selection of files with long filenames.
Dialog Type	Select the type of dialog that you want to present to the user. *Open* allows a user to simply access a file name from the Open dialog. *Save* allows the user to update the contents of the filename selected. *Directory* allows the user to return the path only (no filename) as presented by the Windows dialog.
Don't Report Errors	Check this box to allow errors to be reported if a file name could not be opened or saved.
Return Default on Cancel	Check this box to automatically return the default directory or filename value if the user presses the Cancel button in the appropriate dialog window.
Multi-Select?	Check this box to allow selection of one *or more* files.
Action For Each Selection	Type a valid Clarion language statement to execute for each selected file--typically a procedure call. You may want to pass the **FileName**

Variable as a parameter to the procedure.

The template generates a LOOP to execute the code you specify for each selected file. The generated code reloads the **FileName Variable** with the appropriate filename for each loop cycle.

Assign to Image Control — Check this box if you wish to assign the selected filename's contents to an image control. Use the **Image Control** drop list to select the image control's Field Equate Label.

Resize to control size — Check this box to designate that the filename's contents assigned to the target Image control will be resized to fit the control.

Set Default Mask Value — Press this button to set the default file masks to use when selecting a valid image file to load into the target image control.

File Masks

Use a variable file mask — Check this box to supply the file mask with a variable. This enables the **Variable Mask Value** field to name the variable, and disables the **Mask Description, File Mask,** and **More File Masks** prompts.

Mask Variable — Names the variable that contains the file mask. See *FILEDIALOG* in the *Language Reference* for information on the contents of this variable.

File Mask Description — Type a file type description. The string appears in the drop-down list in the Windows file dialog. You can add additional masks by pressing the **More File Masks** button.

File Mask — Type a file mask specification, such as "*.TXT" or use multiple patterns for this mask separating each with a semicolon, such as "*.BMP;*.GIF". You can add additional masks by pressing the **More File Masks** button.

More File Masks — Press this button to add additional file masks. These masks are available to end the user through the **List files of type** drop-down list in the Windows file dialog.

Window Update Options

Update entire window? — Check this box to refresh the contents of all window controls after the file selection and processing is complete. Clear the box to select specific fields to refresh.

Update Selected Fields — Press this button to select specific fields to refresh after the file selection and processing is complete. The template generates a DISPLAY statement for each field you specify. See *DISPLAY* in the *Language Reference.*

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

DynamicImage

The DynamicImage control provides an image control that must be placed within an existing BrowseGrid control. This template provides the BrowseGrid with the correct image associated with the column specified by the Field to drive image prompt. The image is a clickable image that can perform an action associated with any control on the window. The DynamicImage control requires a BrowseGrid control.

The DynamicImage template provides the following prompts:

Field to drive image	Specifies the column from the table that designates the image file name.

Classes

The classes tab lets you control the class (and object) the template uses. You may accept the default Application Builder Class and it's object (recommended) or you may specify your own or a third party class. Deriving your own class can give you very fine control over the procedure when the standard Application Builder Class is not precisely what you need.

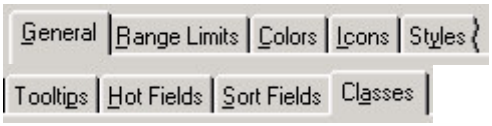
Field Lookup Button control template

The FieldLookupButton template provides an ellipsis (...) button that lets you "look up" the value from a lookup file, such as a state file. CLICK next to an input control to place the lookup button.

The FieldLookupButton template provides the following prompts:

Control with Lookup	Select the associated control for which to perform the lookup by choosing its field equate label from the drop-down list. Typically this is an ENTRY control. The selected control must have an associated lookup procedure. To provide the lookup procedure, RIGHT-CLICK on the control, then choose **Actions** to access its prompts.

File Drop control template



Click on a TAB to see its help

The FileDrop template places a file-loaded scrollable drop-down list on a window. At runtime, the end user can select an item from the list, then assign a value from the selected item's record to a specified target field. You may display one field (such as a description field) but assign another field (such as a code field) from the selected record (see Using drop-down lists to Lookup Records).

Set the DROP attribute to zero (0) to display a list box rather than a drop-down list.

Immediately before you place the FileDrop Control template on your window, the Application Generator prompts you to specify the file to display in the drop-down list. Specify the file in the **Select Field** dialog. You will also need to select a field to serve as the USE variable for the LIST; however, the field you select is only significant if you are displaying one field but assigning another).

Immediately after you place the FileDrop Control template, the Application Generator opens the List Box Formatter so you can specify the fields to display in your list. You may specify the field containing the lookup value as well as other fields from the same or related files.

After you specify your list fields and return to the window under construction, right-click the control, then choose **Actions** from the popup menu to complete the following FileDrop options:

General

Table Schematic Description	Enter a descriptive string that will be displayed in the Table Schematic window for this particular control. This allows you to distinguish one control from another when there are multiple controls populated in a single window.
Field to Fill From	The field in the lookup file whose value is assigned to the Target Field. Press the ellipsis (...) button to select from the **Select Field** dialog.
Target Field	The field that receives the value from the Field to Fill From. Press the ellipsis (...) button to select from the **Select Field** dialog.
More Field Assignments	Press this button to specify additional value assignments from the selected item's record.
Record Filter	Type a valid Clarion expression to limit the contents of the list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter. Filters are generally much slower than Range Limits.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompts.

You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.

Default to first entry if USE variable empty — Check this box to provide an initial default selection--the drop-down list is never initially empty (unless the first file record is a blank one).

IPDRV Options

Do not include Primary key in view — Check this box to specify that primary key components are not projected into the view. This is useful when working with SQL tables, and allows valid GROUP BY SQL statements to be generated. See also SQL Advanced Tab

Range Limits

This tab is only available if you specify a Key for the File in the **Data / Tables Pad** dialog. Because range limits use keys, they are generally much faster than filters.

Range Limit Field — In conjunction with the **Range Limit Type**, specifies a record or group of records for inclusion in the process. Choose a key field on which to limit the records by pressing the ellipsis (...) button.

Range Limit Type — Specifies the type of range limit to apply. Choose one of the following from the drop-down list.

Current Value — Limits the key field to its current value.

Single Value — Lets you limit the key field to a single value. Specify the variable containing that value in the **Range Limit Value** box.

Range of Values — Lets you limit the key field to a range of values. Specify the variables containing the upper and lower limits of the range in the **Low Limit Value** and **High Limit Value** boxes.

File Relationship — Lets you limit the key field to the current value in a related (parent) file. Press the **Related file** ellipsis (...) button to choose the range limiting file. This limits the process to include only those child records matching the current record in the parent file. For example, if your report was a list of Orders, you could limit the process to only those orders for the current Customer.

Colors

This tab is only available if you check the **Color Cells** box in the List Box Formatter. It displays a list of the FileDrop columns which may be colored.

To specify the default colors and any conditional colors, highlight the column's field name, then press the **Properties** button. This opens the **Customize Colors** dialog.

Customize Colors — This dialog lets you specify the default and conditional Foreground and Background colors for normal (unselected) rows; and for selected rows.

Conditional Color Assignments — Below the default colors section is the **Conditional Color Assignments** list. This list lets you set colors to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the **Insert** button.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompts.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Icons

This tab is only available if you check the **Icons** box in the List Box Formatter. It displays a list of the FileDrop columns which can display icons.

To specify default icons and any conditional icons, highlight the column's field name then press the **Properties** button. This opens the **Customize Icons** dialog.

Customize Icons — This dialog lets you specify the default icon and conditional icons for the FileDrop column.

Default Icon — The default icon to display. Type the icon (.ICO) filename. You can also name a variable to use as the default icon, using the !*variable* format. The variable may be a string type that stores the icon filename, or check the **The expression is a number** check box if you are referencing a previously named icon assigned to the IconList property.

Conditional Icon Usage — Below the **Default Icon** section is the **Conditional Icon Usage** list. This list lets you set icons to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated icon, press the **Insert** button.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Styles

This tab is only available if you check the **Style** box in the List Box Formatter. It displays a list of the List Box columns that may have applied styles.

To specify the default styles and any conditional styles, highlight the column's field name, then press the Properties button. This opens the Customize List Box Styles dialog. A default style may also be defined on the List Box Formatter Appearance tab.

Create GreenBar Effect Check this box to create a GreenBar effect (alternating colors on each row) on your list box. You will be prompted to select two styles to use that represent the appearance of each alternating row

Default Style This entry lets you specify the default style to be used for the column.

Conditional Styles This list lets you define the styles to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the Insert button.

Condition Provide a valid Clarion expression that when evaluates to true (nonzero or non-blank) will cause the Style to be applied.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Style Define the style number that will be applied to the column when the Condition evaluates to true (nonzero or non-blank).

At run-time the expressions are evaluated, and the styles for the first true expression are used.

All Styles must be defined. For more information on defining styles see FORMAT (set LIST or Combo layout).

Tooltips

This tab is only active if you check the **Tooltip** box in the List Box Formatter. It displays a list of the List Box columns which may have applied tool tips. Press the **Properties** button to display the *Customize BrowseBox Tooltips* dialog.

Tooltip variable

You can specify a default tool tip string value in the List Box Formatter. This entry lets you override the default tooltip to use a value contained in a variable. Press the ellipsis button to select a variable that will contain the text of your column's tool tip.

Hot Fields

When you select the Hot Fields tab, you can specify fields not populated in the list to add to the QUEUE. When scrolling through the file, the generated source code reads the data for these fields from the QUEUE, rather than from the disk. This speeds up list box updates.

Specifying "Hot" fields also lets you place controls outside the FileDrop that are updated whenever a different record is selected in the list box. Elements of the Primary Key and the current key are always included in the QUEUE, so they do not need to be inserted in the Hot Field list.

This dialog also lets you BIND a field. You must BIND any field, variable, or EQUATE that is used in a filter expression or as a field to total. If the field you are BINDing is not in the VIEW, check the **Not in View** checkbox.

Sort Fields

This tab lets you add fields by which the items in the drop-down list are sorted. The sort fields are in addition to any Key specified for the FileDropCombo. Press the **Insert** button to add fields to the list.

Classes Tab

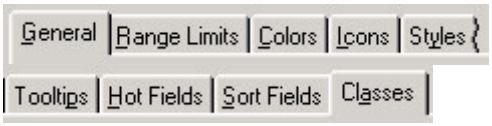
Use the Classes tab to override the global settings for the Class.

Other Prompts

The List Properties for this control are the same as for a list; however, the following prompts may require some additional explanation:

Use	Takes either a field equate label, or the label of a variable to receive the value from the first field populated in the list. In the FileDrop Control template context, this functionality is replaced by the more flexible Target Field setting.
From	This field defaults to Queue:FileDrop. Queue:FileDrop is the label of the QUEUE the template uses to fill the list. Typically, you should not change this value.
Mark	Takes the label of the Queue:FileDrop:Mark QUEUE field to allow the user to select more than one item from the list. The Queue:FileDrop:Mark field contains 1 for selected items and 0 for unselected items.

File Drop Combo control template



Click on a TAB to see its help

The FileDropCombo template generates code to display a data file in a scrollable list, select one of the records from the list, then assign a value from the selected record to a specified target field. Note that you may display one field (such as a description field) but assign another field (such as a code field) from the selected record (see Using drop-down lists to Lookup Records). Also, because the template is based on a COMBO control, the generated code accepts entry values that may not exist in the displayed list and optionally adds these new values to the lookup file.

Immediately before you place the FileDropCombo Control template on your window, the Application Generator prompts you to specify the file to display in the drop-down list. Specify the file in the **Select Field** dialog. You will also need to select a field from the file to serve as the USE variable for the COMBO. The USE variable is significant when you Allow Updates from the FileDropCombo or when you display one field but assign another.

Immediately after you place the FileDropCombo Control template, the Application Generator opens the List Box Formatter so you can specify the fields to display in your list. You may specify the field containing the lookup value as well as other fields with associated information.

After you specify your list fields and return to the window under construction, RIGHT-CLICK the control, then choose **Actions** from the popup menu to complete the following FileDropCombo options:

General

Table Schematic Description	Enter a descriptive string that will be displayed in the Table Schematic window for this particular control. This allows you to distinguish one control from another when there are multiple controls populated in a single window.
Field to Fill From	The field in the lookup file whose value is assigned to the Target Field. Press the ellipsis (...) button to select from the **Select Field** dialog.
Target Field	The field that receives the value from the Field to Fill From. Press the ellipsis (...) button to select from the **Select Field** dialog.
More Field Assignments	Press this button to specify additional value assignments from the selected item's record.
Record Filter	Type a valid Clarion expression to limit the contents of the list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter. Filters are generally much slower than Range Limits. You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.
Default to first entry if USE variable empty	Check this box to provide an initial default selection--the drop-down list is never initially empty (unless the first file record is a blank one).

IPDRV Options	Press this button if you are using the IP Driver in your application, and need to adjust the default settings in the IPDRV Options dialog.
Do not include Primary key in view	Check this box to specify that primary key components are not projected into the view. This is useful when working with SQL tables, and allows valid GROUP BY SQL statements to be generated.
Remove duplicate entries	Check this box to remove duplicates from the list.
Keep View synchronized with Selection?	Check this box to update the VIEWs record buffers to match the selected item.
Automatic Entry Field Completion	Check this box to allow the entry control to act as a incremental locator. As you enter characters, the template locates and populates the closest matching record in the list box (equal to or greater than the current entry field's contents).
Case Sensitive matches?	Check this box to consider case when matching entered values with values in the lookup file.

Range Limits

This tab is only available if you specify a Key for the File in the **Data / Tables Pad** dialog. Because range limits use keys, they are generally much faster than filters.

Range Limit Field
In conjunction with the **Range Limit Type**, specifies a record or group of records for inclusion in the process. Choose a key field on which to limit the records by pressing the ellipsis (...) button.

Range Limit Type
Specifies the type of range limit to apply. Choose one of the following from the drop-down list.

Current Value
Limits the key field to its current value.

Single Value
Lets you limit the key field to a single value. Specify the variable containing that value in the **Range Limit Value** box.

Range of Values
Lets you limit the key field to a range of values. Specify the variables containing the upper and lower limits of the range in the **Low Limit Value** and **High Limit Value** boxes.

File Relationship
Lets you limit the key field to the current value in a related (parent) file. Press the **Related file** ellipsis (...) button to choose the range limiting file. This limits the process to include only those child records matching the current record in the parent file. For example, if your report was a list of Orders, you could limit the process to only those orders for the current Customer.

Colors

This tab is only available if you check the **Color Cells** box in the List Box Formatter. It displays a list of the FileDropCombo columns which may be colored.

To specify the default colors and any conditional colors, highlight the column's field name, then press the **Properties** button. This opens the **Customize Colors** dialog.

Customize Colors

This dialog lets you specify the default and conditional Foreground and Background colors for normal (unselected) rows; and for selected rows.

Conditional Color Assignments
Below the default colors section is the **Conditional Color Assignments** list. This list lets you set colors to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the **Insert** button.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Icons

This tab is only available if you check the **Icons** box in the List Box Formatter. It displays a list of the FileDropCombo columns which can display icons.

You can also name a variable to use as the default icon, using the !*variable* format. The variable may be a string type that stores the icon filename, or check the **The expression is a number** check box if you are referencing a previously named icon assigned to the IconList property.

To specify default icons and any conditional icons, highlight the column's field name then press the **Properties** button. This opens the **Customize Icons** dialog.

Customize Icons

This dialog lets you specify the default icon and conditional icons for the FileDropCombo column.

Default Icon
The default icon to display. Type the icon (.ICO) filename. To specify a variable icon name, prefix the variable name with an exclamation point (i.e., *!MyIconVariable*)

Conditional Icon Usage
Below the **Default Icon** section is the **Conditional Icon Usage** list. This list lets you set icons to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated icon, press the **Insert** button.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Styles

This tab is only available if you check the **Style** box in the List Box Formatter. It displays a list of the List Box columns that may have applied styles.

To specify the default styles and any conditional styles, highlight the column's field name, then press the Properties button. This opens the Customize List Box Styles dialog. A default style may also be defined on the List Box Formatter Appearance tab.

Create GreenBar Effect

Check this box to create a GreenBar effect (alternating colors on each row) on your list box. You will be prompted to select two styles to use that represent the appearance of each alternating row

Default Style

This entry lets you specify the default style to be used for the column.

Conditional Styles

This list lets you define the styles to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the Insert button.

Condition
Provide a valid Clarion expression that when evaluates to true (nonzero or non-blank) will cause the Style to be applied.

Style
Define the style number that will be applied to the column when the Condition evaluates to true (nonzero or non-blank).

At run-time the expressions are evaluated, and the styles for the first true expression are used.

All Styles must be defined. For more information on defining styles see FORMAT (set LIST or Combo layout).

Tooltips

This tab is only active if you check the **Tooltip** box in the List Box Formatter. It displays a list of the List Box columns which may have applied tool tips. Press the **Properties** button to display the *Customize BrowseBox Tooltips* dialog.

Tooltip variable

You can specify a default tool tip string value in the List Box Formatter. This entry lets you override the default tooltip to use a value contained in a variable. Press the ellipsis button to select a variable that will contain the text of your column's tool tip.

Update Behavior

This tab lets you use the entry portion of the COMBO to initiate adding a new record to the lookup file. If the user types a value in the entry box that is not already in the list, the generated code can add a new record directly, or it can call a separate procedure to add the new entry.

Allow Updates

Clear this box to only allow items in the drop list to be entered as as valid entry. Values not matching the target field in the drop list will be automatically cleared.

Check this box to add new entries to the lookup file, and to enable the Update Procedure prompt. If the Update Procedure prompt is left blank, an entry not in the drop list will pop up a window that contains the following message:

"Record Match not found, do you wish to add a new one?"

If you press the "Yes" button, the new value will be accepted, and added to the lookup file. If you press the "No" button, the entry will be cleared and not added to the lookup file.

If you wish to allow unvalidated entries (entries NOT contained in the lookup file), use a standard File Combo control.

Update Procedure

Name the procedure to call to add the new record, or leave this field blank if no update procedure is needed.

No update procedure is needed for lookup files with only one required field (the field specified by the COMBO's USE variable). Non-USE fields are CLEARed, unless range limited or auto-incremented.

Silent Automatic Add	This option is used is you wish to allow "silent adds" to the drop combo list box, without confirmation or display of an update procedure. This option is only enabled if you have checked the **Allow Updates** box, but have not entered an **Update Procedure**.

Hot Fields

Use the Hot Fields tab to specify fields to add to the QUEUE that are not displayed in the list. When scrolling through the file, the generated source code reads the data for these fields from the QUEUE, rather than from the disk. This speeds up list box updates.

Specifying Hot Fields effectively lets you update other controls whenever a new record is selected in the list box. Elements of the Primary Key and the current key are always included in the QUEUE, so they do not need to be inserted in the Hot Field list.

Press the **Insert** button to add fields to the list.

If the field you are BINDing is not in the VIEW, check the **Not in View** checkbox.

Sort Fields

This tab lets you add fields by which the items in the drop-down list are sorted. The sort fields are in addition to any Key specified for the FileDropCombo. Press the **Insert** button to add fields to the list.

Classes Tab

Use the Classes tab to override the global settings for the Class. See Classes Tab.

Other Prompts

The List Properties for this control are the same as for a list; however, the following prompts may require some additional explanation:

Use	Takes either a field equate label or the label of a variable to receive the value from the first field populated in the list. In the FileDropCombo Control template context, the assignment functionality is replaced by the more flexible Target Field; however, the USE variable is significant when you Allow Updates from the FileDropCombo (see *Update Behavior* for more information).
From	This field defaults to Queue:FileDropCombo. Queue:FileDropCombo is the field equate label of the QUEUE the template generates to fill the list. Typically, you should not change this value.
Mark	Takes the label of the Queue:FileDropCombo:Mark QUEUE field to allow the user to select more than one item from the list. The Queue:FileDropCombo:Mark field contains 1 for selected items and 0 for unselected items.

FormVCRButtons Control Template

The FormVCRButtons Control Template is used to navigate to different records (or rows) from a single Form window. If desired, you can also perform standard database operations on selected records. It is designed as an alternative to the standard Browse-Form paradigm.

Note:

This template is an alternative *replacement* of the browse that is used to call a Form. Currently this template is only available in the ABC template chain.

The Form VCR buttons control template provides the following functions:

Navigation:

Go to the first record

Page up to the record

Go to the Prior Record

Go to the Next Record

Page Down to the Record

Go to the Last Record

Actions:

Switch form to View-Only mode

Switch the form to Recursive Add Mode. The OK button on the form saves the record and allows you to insert another. The Cancel button cancels the operation and returns to the calling procedure.

Switch the form to change mode. The OK button on the form saves the record and allows you to navigate to another record. The Cancel button cancels the operation and returns to the calling procedure.

Switch the form to delete mode. The OK button on the form saves the record and allows you to navigate to another record. The Cancel button cancels the operation and returns to the calling procedure.

To make this control template active (or visible) to the Form procedure, you must have a Save Button control template populated.

The FormVCRButtons template is designed to merge the standard functionality of the Browse Box template with a standard form. Moreover, it is actually designed as an *alternative* to the Browse-Form paradigm (e.g., calling a form from a menu).

In addition to these functions, the FormVCRButtons template provides these additional prompts:

Default Behavior tab:

Default Action

Select from the drop list the default mode to set when the Form window is first opened.

Do not include Primary key in view

Check this box to specify that primary key components are not projected into the view. This is useful when working with SQL tables, and allows valid GROUP BY SQL statements to be generated.

Change to Insert mode if table is empty

Check this box to switch the default behavior to Insert if the primary file specified by the update procedure is empty. The only reason that you would not check this box is the case where Inserts are not allowed on the Form.

Page Size

Enter a number (manually or with the spin control) to designate how many records you wish to Page Up and Down when navigating. The default value is 10.

If you need to restrict a particular action supported by the FormVCRButtons (i.e., Insert, Change, Delete) you can simply delete the appropriate button.

In order to enable locator activity with the FormVCRButtons template, you must specify a key for the Form's primary file.

The default behavior for this template is the same as a standard Browse Box. For more help on the other Default Behavior prompts (Locators, Range Limit, Record Filters, etc.), see Browse Box - Default Behavior

FormVCRButtons – Conditional Behavior

The Conditional Behavior prompts of the FormVCRButtons control template are identical to the Browse Box conditional behavior. See Browse Box - Conditional Behavior.

FormVCRButtons – Hot Fields

The Hot Fields prompts of the FormVCRButtons control template are identical to the Browse Box Hot Fields. See Browse Box - Hot Fields.

FormVCRButtons – Classes

The Classes prompts of the FormVCRButtons control template are identical to the Browse Box Classes options. See See Classes Tab.

FrameBrowseControl

The *FrameBrowseControl* places thirteen (13) standard command buttons on the toolbar of an MDI APPLICATION (*Frame*). When the user presses these buttons, the template generated code posts appropriate events (scroll up, scroll down, add, change, delete, help, etc.) to the active procedure.

You may delete buttons that your application does not use. For example, the standard templates by default do not use the locate button.

The buttons are designed to work with the BrowseBox Control template, the RelationTree Control template, and the FormVCRControls Extension template; that is, the buttons remain disabled until the program calls a procedure with a BrowseBox template or a RelationTree template whose **Accept browse control from Toolbar** box is checked, or the BrowseBox procedure calls a Form procedure with a FormVCRControls extension template.

In addition, the called procedure's WINDOW must have the MDI attribute--the standard *Browse* and *Form* templates declare MDI windows by default--you don't need to do anything special to accomplish this. The *BrowseBox* and *RelationTree* templates also checks the **Accept browse control from Toolbar** box by default--so again, you don't need to do anything special to accomplish this.

The *FrameBrowseControl* toolbar buttons operate as follows:

Scrolls to the first row in a BrowseBox or to the previous parent record in a RelationTree. For Form procedures, saves the current record before scrolling.

Scrolls up one page in a BrowseBox or to the previous record on the same level in a RelationTree. For Form procedures, saves the current record before scrolling.

Scrolls up one row in the BrowseBox or to the previous record on any level in a RelationTree. For Form procedures, saves the current record before scrolling.

Locates a specific record in a BrowseBox. See *Control Templates--BrowseBox* for information on specifying these locators.

Scrolls down one row in the BrowseBox or to the next record on any level in a RelationTree, expanding the tree branch if necessary. For Form procedures, saves the current record before scrolling.

Scrolls down one page in the BrowseBox or to the next record on the same level in a RelationTree. For Form procedures, saves the current record before scrolling.

Scrolls to the last row in the BrowseBox or to the next parent record in a RelationTree. For Form procedures, saves the current record before scrolling.

Selects the highlighted row in a BrowseBox.This is only appropriate when the Procedure is called to select a record. For example, when called as a lookup.

For a BrowseBox, calls a *Form* procedure to add a new record. For a RelationTree, calls a *Form* procedure to add a child record of the currently highlighted record. For a Form procedure, adds another record of the same type.

Calls a *Form* procedure to change the record highlighted in the BrowseBox or RelationTree.

Deletes the record highlighted in the BrowseBox or RelationTree. The BrowseBox delete behavior is determined by the settings on the Update Buttons Control template.

On a *Form* procedure only, pastes into the field with focus, the corresponding value from the previously processed record (the value in the record buffer). In other words, repeat the value from the previous saved record. Also known as the "ditto" button.

Invokes Windows standard help behavior: calls WINHELP.EXE with the help topic or keyword specified by the WINDOW's HLP attribute.

The Properties dialogs for the *FrameBrowseControl* buttons is the normal **Button Properties** dialog.

Help Button control template

The Help Button control template populates a help button on a window that calls the program's help file when pressed. The help topic default is the Help ID that you have defined for the window.

There are no special prompts or actions for this control template.

IPDRV Options

This dialog is useful if you are using the IP Driver addon product, and accessible through the **Browse Box Procedure Properties > Browse Box Behavior > Extended Options**.

The IP Driver is used for data access and transfer over an IP protocol. More information regarding this powerful addon can be found at the SoftVelocity web site.

Use MRP (Multiple Request Packet)

The IP Driver Version 2.0 introduced the use of Multiple Request Packets (MRP). MRPs are designed to improve the performance of data transfer via IP protocol, by packaging multiple requests (records) into a single IP data packet.

If MRP use is set to ON in a BROWSE procedure, the IP Driver returns a full page of records from the server, instead of one record at a time. In a Report or Process, the MRP is set to the value of RecordsPerCycle. The MRP reduces the roundtrips to the server dramatically. However, if you are doing hand-coded file access for each record read (i.e., NEXT, GET, ADD, PUT, etc.) inside the LOOP structure used to fill the List control, or during a Report or Process Procedure, set this checkbox to FALSE (unchecked).

For more information, see the following related properties and methods:

UseMRPUseMRP_IPD_Multiple_Request_Packet_Support_, SetUseMRPSetUseMRP_set_UseMRP_value_BC, GetUseMRPGetUseMRP_get_UseMRP_value_BC

MultiChildRelationTree control template

Click on a TAB to see its help

The MultiChildRelationTree control extends the standard RelationTree control with a number of new enhancements and developer control. Like the standard RelationTree, the MultiChildRelationTree control is a list box formatted to display as a collapsible hierarchical list. This Control template provides an alternative for the Browse-Form paradigm. A single MultiChildRelationTree control can replace several Browse-Form combinations.

Using this template, you can specify multiple related files to display on multiple levels (up to 29) of a hierarchical list--*with an associated update procedure for each level*. The related files are declared in the Data / Tables Pad--the Primary (Parent) file and multiple chains of related secondary Child files (Parent-Child-GrandChild). But more so, **This template allows multiple child tables to be attached to a single primary record. See the shipping example for an illustration of this.**

The MultiChildRelationTree template employs a fully-loaded QUEUE for the root level. The child levels are demand-loaded when a branch is expanded.

This template is not appropriate for databases with a very large primary file. For large files you should use the BrowseBox Control template.

The plus (+) sign indicates a collapsed level that expands when the user CLICKS on the plus (+) sign. Conversely, the minus (-) sign indicates an expanded level that collapses when the user CLICKS on the minus (-) sign.

To register the MultiChildRelationTree Control template:

Access the Template Registry from the IDE Tools menu. In the Template Registry window, press the **Register** button, and locate and select the MCRTREE.TPL file.

To create a tree using the MultiChildRelationTree Control template:

Place a **MultiChildRelationTree** Control template on a window.

This opens the **List Box Formatter**. Use the **List Box Formatter** to enable colorization, icon display, or horizontal scrolling in your tree control (see *The List Box Formatter*). Do *not* use the **List Box Formatter** to populate fields in the tree control.

The tree control is a single column list, therefore you must specify a column scroll bar rather than a list scroll bar to accomplish horizontal scrolling.

Press the **OK** button on the **List Box Formatter**.

RIGHT-CLICK on the RelationTree Control template and choose **Actions** from the popup menu.

Open the **Data / Tables Pad** to specify the file schematic for the control.

Specify the Primary (Parent) file and optional chains of related Secondary Child files (Parent-Child-GrandChild).

Complete the MultiChildRelationTree template prompts.

The MultiChildRelationTree template provides the following prompts:

General Tab Details

Table Schematic Description
Enter a descriptive string that will be displayed in the Table Schematic window for this particular control. This allows you to distinguish one control from another when there are multiple controls populated in a single window.

Accept control from Toolbar
Check this box to accept navigation events and other relation tree control events generated by the *FrameBrowseControl* control template on the APPLICATION's toolbar. See *FrameBrowseControl* for more information on these toolbar buttons and their operation. Clear this box to disable the *FrameBrowseControl* toolbar buttons for this procedure.

Enable Tree Root
Check this box to create a root (Single top branch). All primary file records will be displayed directly below this root. More options for the Root branch is found on the template Root tab.

Disable Primary Table Level
Check this box to remove the primary table from the template interface and allow the creation of the tree using only the Secondary (child) tables, in order to create a "multi-parent" tree. This is very handy when a dummy table is used as the primary table and you don't want to show any Root node based on this table.

Tree Line Height
Enter a number in dialog units (unless PROP:Pixels is active) to set the line height of each row generated in the relation tree. This option sets the PROP:LineHeight property.

Expand Key
Specify a keystroke to expand the selected list item--display its children. Press the ellipsis button (...) to select special keys such as ESC, TAB or ENTER. See *Controls and Their Properties--Common Control Attributes--Setting the KEY Attribute* for more information on this dialog.

Contract Key
Specify a keystroke to contract the selected list item--hide its children. Press the ellipsis button (...) to select special keys such as ESC, TAB or ENTER. See *Controls and Their Properties--Common Control Attributes--Setting the KEY Attribute* for more information on this dialog.

Initial State
Select Expanded or Contracted from the drop list options. This is the initial state of the relation tree as the window is first opened.

Select from the following **Popup** options:

Expanding/Contracting via popup
Specify the RIGHT-CLICK popup menu for the RelationTree includes "Expand All" and "Contract All" commands.

Use One Item for Both Actions
This box is enabled if you have enabled the **Expanding/Contracting via popup** option. Check this box to allow one popup item to toggle the Expanding/Contraction action, or leave blank to create distinct menu items for each action.

Expand/Contract Popup priority
Specify a number relative to the other selections below that will determine the *order* that this popup menu item will appear in the structure. A value of 1 would designate it as the first item.

Refresh Via Popup
Check this box to add a refresh option to the popup menu.

Refresh Popup Priority
Specify a number relative to the other selections below that will determine the *order* that this popup menu item will appear in the structure. A value of 1 would designate it as the first item.

Root Tab Options

The options listed here are only made available if you had checked the **Enable Tree Root** option in the template's **General** tab.

Tree Heading Text
An optional text heading at the top of the tree. Tree Heading Text is required to let the user add a record at the root level. Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Tree Heading Icon
An optional icon at the top of the tree. Icons must be enabled in the List Box Formatter for this prompt to be enabled.

Colors Sub Tab

If you have enabled the **Colors** option in the List Box Formatter, you can specify colors for the Root node as follows:

Default Colors

This dialog lets you specify the default and conditional Foreground and Background colors for normal (unselected) and selected columns.

Conditional Colors
Below the default colors section is the **Conditional Color Assignments** list. This list lets you set colors to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the **Insert** button.

At run-time the expressions are evaluated, and the colors for the first true expression are used.

Styles Sub Tab

This tab is only available if you check the **Style** box in the List Box Formatter. It displays a list of the BrowseBox columns that may have applied styles.

To specify the default styles and any conditional styles, highlight the column's field name, then press the Properties button. This opens the Customize BrowseBox Styles dialog. A default style may also be defined on the List Box Formatter Appearance tab.

Default Style

This entry lets you specify the default style to be used for the column.

Style Type

Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number

Based on the Style Type selected, select a style's description or number from the drop list control.

Conditional Styles

This list lets you define the styles to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the Insert button.

Condition
Provide a valid Clarion expression that when evaluates to true (nonzero or non-blank) will cause the Style to be applied.

Style Type
Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number
Based on the Style Type selected, select a style's description or number from the drop list control.

At run-time the expressions are evaluated, and the styles for the first true expression are used.

All Styles must be defined. For more information on defining styles see Listbox Styles.

Primary Level Tab – Primary File Details

Level Title

The text to display for the primary file level. This may be any valid Clarion expression, for example:

```
!CLIP(CUST:LastName)&' '&CUST:FirstName
```

Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt. All expressions must be prepended with an exclamation point as shown above.

Field to Display

The field name to display for the primary file level. This may also be any valid Clarion expression. Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Advanced Display

If the text to display is part of a more complex expression, or even perhaps the result of a procedure call, this button opens an appropriate embed point to use to construct the value to return.

Record Filter
Type a valid Clarion expression to limit the contents of the list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter.

You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.

There is also **Range Limit** support in the MultiChild Relation Tree. Set the Range Limit options as you would in a standard Browse Box. Click here for more information.

Icons

This tab is only available if you select an **Icon** in the **Appearance** tab in the List Box Formatter.

Use different icons for Expand and Contract
Check this box to enable you to select different icons to display when the relation tree is in an expanded or contracted state. This option also cascades to the *Conditional Icons* option.

Default Icon
To specify the default icon for the primary file display string, first select the type of icon to use.

Icon Type
In the drop list provided, select from *Built-in* (pick from a built-in IDE list), *File* (a standard .ICO file), *Variable* (using the *!variable* format), or *External* (to reference icons in a composite *ICO file, using the *filename.ico[x]* format).

Conditional Icons
This dialog lets you specify conditional icons for the primary file display string.

To specify conditional icons for the primary file display string, press the **Insert** button. This opens the **Conditional Icon Usage** dialog.

Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Condition
Type a valid Clarion expression to evaluate at runtime.

Icon Type and Icon
In the drop list provided, select from *Built-in* (pick from a built-in IDE list), *File* (a standard .ICO file), *Variable* (using the *!variable* format), or *External* (to reference icons in a composite *ICO file, using the *filename.ico[x]* format).

At run-time these conditions are evaluated, and the icon for the first true condition in the list is used.

Colors Tab

This tab is only available if you check the **Colors** box in the **Flags** area located in the *List Box Formatter*.

Default Colors
To specify the default colors for the primary file display string, type color EQUATEs (from \LIBSRC\EQUATES.CLW) in the entry fields or press the ellipsis (...) buttons to select colors from the *Select Color* dialog.

Conditional Colors
To specify conditional colors for the primary file display string, press the **Insert** button. This opens the **Conditional Color Assignments** dialog.

Condition
Type a valid Clarion expression to evaluate at runtime, then type color EQUATEs (from \LIBSRC\EQUATES.CLW) in the entry fields or press the ellipsis (...) buttons to select colors from the **Select Color** dialog.

Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

At run-time these conditions are evaluated, and the colors for the first true condition in the list are used.

Styles Tab

This tab is only available if you check the **Style** box in the List Box Formatter. It displays a list of the BrowseBox columns that may have applied styles.

To specify the default styles and any conditional styles, highlight the column's field name, then press the Properties button. This opens the Customize BrowseBox Styles dialog. A default style may also be defined on the List Box Formatter Appearance tab.

Default Style

This entry lets you specify the default style to be used for the column.

Style Type

Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number

Based on the Style Type selected, select a style's description or number from the drop list control.

Conditional Styles

This list lets you define the styles to apply when an expression evaluates to true (nonzero or non-blank). To add an expression and its associated colors, press the Insert button.

Condition
Provide a valid Clarion expression that when evaluates to true (nonzero or non-blank) will cause the Style to be applied.

Style Type
Use the drop list to select *Local List* or *Style Number*. The *Local List* displays the descriptions of the styles that you created in the Listbox Styles dialog. You can also reference the styles by *Style Number* only.

Style or Style Number
Based on the Style Type selected, select a style's description or number from the drop list control.

At run-time the expressions are evaluated, and the styles for the first true expression are used.

All Styles must be defined. For more information on defining styles see Listbox Styles.

Hot Fields Tab

The options in this tab are identical to the standard Hot Fields dialog found in the Browse Box control. Any field that needs to be accessed individually within a filter or in a call to any procedure from this template must be included in the Hot Fields list.

Child Levels Settings

The Child Levels settings are identical to the primary file settings. Highlight the secondary file, then press the **Properties** button below the **Secondary Files** list box. See *RelationTree Overview* for information on how to specify the secondary files with the *Select File* dialog.

Select from the following options:

Level Title

The text to display for the child table level. Expressions are also valid here. Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in this prompt.

Field to Display

The field name to display for the primary file level. This may also be any valid Clarion expression. Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Advanced Display

If the text to display is part of a more complex expression, or even perhaps the result of a procedure call, this button opens an appropriate embed point to use to construct the value to return.

Custom Join

A powerful feature of this template is that you can use tables that are not necessarily related in the dictionary to build your relation tree. If you have specified a Custom Join in the table schematic for this table, you can specify the type of join used here by pressing this button. In the *Custom Join* dialog you select the Key and Key Field used, and then specify the join criteria. The Join of this table to the primary table can be based on a *Clear Value*, a *Fixed Value*, or a *Variable Value*.

For example:

Assume that a *User* and *Computers* table are not related to any table in the dictionary.

You can show them at the same tree level by creating a "Dummy" root:

```
DummyRoot
|
Users
|   \---ComputersAssigned
|                       \---ComputersAlias
|
Computers
```

The computer is assigned to a user with a matching UserId and ComputerId

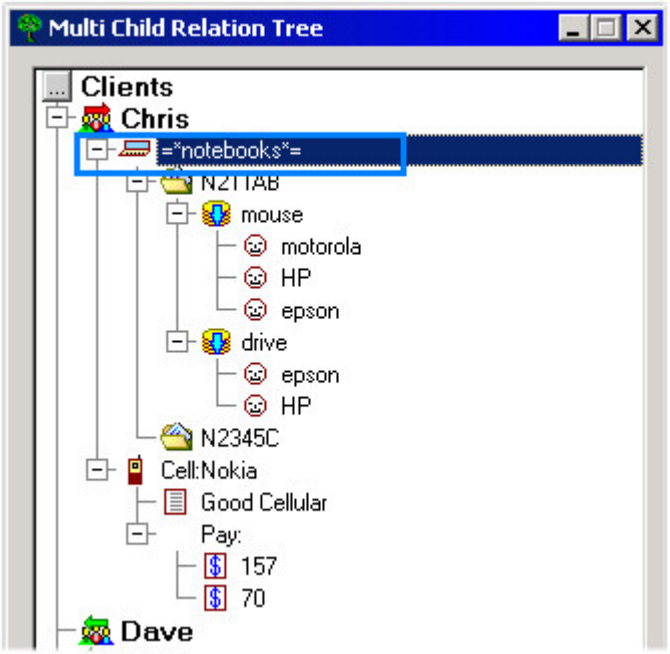
Users and Computer are *not* be related to DummyRoot. The custom join will use the "Clear Value" option to show *all* Users and *all* Computers.

In the case of *ComputersAssigned* and *ComputersAlias*, the template uses smart logic to show them all like one entry because the relation between ComputersAssigned and ComputersAlias is a Many to One.

Record Filter
Type a valid Clarion expression to limit the contents of the list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter.

You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.

Level Root
An optional text heading can be added to each child table to display at the top of the child node. For example:



Press the ellipsis button to call the *Select Column* dialog. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

The **Level Root Format** button opens an additional dialog to control the Colors, Icons and Styles used in the Child Root. The options are identical to the Primary Root dialog.

RelationTree Embed Points

The RelationTree Control template provides a comprehensive set of embed points to allow full customization of the control's behavior.

MultiChildRelationTree Button control template

The **MultiChildRelationTree Button** control template is used with the **MultiChildRelationTree** control template to extend the options of any selected tree element. The idea behind this template is to specify a *single* button which in turn executes source from a *multiple* embed points based on the active secondary child table selected.

General Tab

Disable Condition
Specify a condition that will disable the button for all tree elements when evaluated to TRUE. An example of this is an "access level" that grant rights for special modifications specified by this button.

Use Button via Popup
Check this box to allow this button (when enabled) to appear on the relation tree popup menu.

Add Separation
Check this box to add a popup menu separator after the button item.

Popup Text
You can use the default button text in the popup menu, or specify via the **Use Alternative Text** check box different popup menu text here.

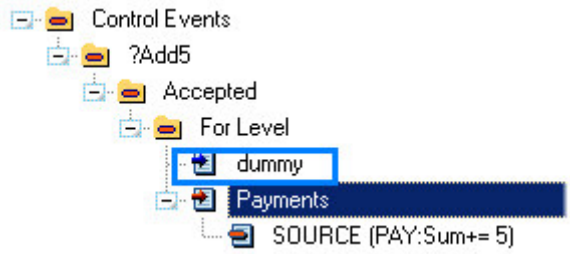
Popup Priority
The order of the popup menu item is specified by a priority number here. It is recommended to develop standards of priorities to prevent clashes or duplicates with other Relation Tree templates.

Refresh Tree After Action
After the button action is performed, check this box to force a refresh of the relation tree's contents.

Primary Level Tab

Item Options:

Enable Button
Check this box to enable this button when a primary record is active (selected). The action of this button will be controlled by the active source code of this primary table's button embed point. For example, if the primary table is called "Dummy", you will see the following embed point become available the next time you enter the Embed window:



Disable Condition

Specify an optional condition that will disable the button for the selected primary record when evaluated to TRUE.

Allow Popup
Check this box to allow this button (when enabled) to also appear on the relation tree popup menu when a primary record is selected.

Root Options:

Enable Button on Root
Check this box to enable this button when the primary root element is active (selected).

Disable Condition
Specify an optional condition that will disable the button for the selected primary root when evaluated to TRUE.

Allow Popup on Root
Check this box to allow this button (when enabled) to also appear on the relation tree popup menu when the primary root is selected.

Child Levels Tab

The list box shown on this tab will list all child tables used in the relation tree. The following naming convention is used for each list box item.

Item: (*ED*)-(*P*) Root (*ED*)-(*P*) for <child table name>

where *E* – Enabled, *D* – Disabled, *P* – Popup Enabled

Item Options:

Enable Button
Check this box to enable this button when the child element is active (selected).

Disable Condition
Specify a condition that will disable the button for the selected child element when evaluated to TRUE.

Allow Popup
Check this box to allow this button (when enabled) to appear on the relation tree popup menu.

Root Options:

Enable Button on Root
Check this box to enable this button when the secondary (child) root element is active (selected).

Disable Condition
Specify an optional condition that will disable the button for the selected secondary (child) root when evaluated to TRUE.

Allow Popup on Root
Check this box to allow this button (when enabled) to also appear on the relation tree popup menu when the secondary (child) root is selected.

MultiChildRelationTree Expand/Contract Buttons control template

This Control template adds two buttons (**Expand** and **Contract**) which let the user expand or contract all the items in the relation tree. A MultiChild Relation Tree control template. must be present on the window. There are no prompts for this template.

MultiChildRelationTree Select Button control template

This Control template adds a **Select** button which lets the user select a record based on the selected branch of the relation tree. A MultiChild Relation Tree control template. must be present on the window. There are no prompts for this template.

MultiChildRelationTree Update Buttons control template

This Control template is dependent upon the existence of the MultiChildRelationTree control template, and adds three buttons (**Insert**, **Change**, and **Delete**) which allow the user to call the associated update procedure for the selected level of the relation tree (if an update procedure has been specified) . The Change and Delete buttons correspond to the currently highlighted record. The Insert button adds the target child record (the next level down the tree structure).

The following template prompts are available:

General Tab

The options in the General tab target the relation tree's popup options.

Use Update via Popup

Check this box to allow update actions (Insert, Change, Delete) to be called from a popup menu

Add Separation

Check this box to also include a separator in the popup menu after these update items

Use Button Text for Popup (Text)

Check this box to automatically assign the update button text to the popup menu item. If you turn this option off, the default text of the popup menu (Insert…, Change…, Delete) will be used.

Use Alternative Update (Key)

If you would like to assign an alternate key to call the popup menu, check this box, and designate an Activation Key. You can use the ellipsis button to access the *Select Key* dialog to assist you.

Use Help Tool tip

Check this box if you would like to use the help tool tip of each update button as a tool tip for the popup menu.

Popup Priority

Enter a number here that will set the order that the update items will appear on the popup menu. A smaller number will bring the items to the top of the popup menu.

Primary Level

The options in the Primary Level tab target the optional Update Procedure used for the relation tree's primary table.

Update Procedure
Select the update procedure to call for the primary file. You can enter a new one, or select from the drop list. The procedure selected may be accessed with the RIGHT-CLICK popup menu when enabled automatically provided when you specify an update procedure. The default popup menu text is "Insert…," "Change…," and "Delete." If your update procedure has parameters, you can specify them in the **Call with parameters** option.

Finally, optionally check the **Allow Insert**, **Allow Change**, **Allow Delete**, or **Allow View** check boxes to enable the appropriate action.

Child Levels

The list box in the Child Levels tab displays all child tables used in the relation tree. If an update procedure is designated for a child table, the list box item will show "Edit <Child Table> with <update procedure>". If am update procedure is not designated, the list box item will show "<child table> not edited" instead. Press the **Properties** button to access the *Secondary Level* dialog and the following options:

Update Procedure
Select the update procedure to call for the selected child table. You can enter a new one, or select from the drop list. The procedure selected may be accessed with the RIGHT-CLICK popup menu when enabled automatically provided when you specify an update procedure. The default popup menu text is "Insert…," "Change…," and "Delete." If your update procedure has parameters, you can specify them in the **Call with parameters** option. An optional description can be added here, which is only used for developer documentation (it is not generated in source).

Finally, optionally check the **Allow Insert**, **Allow Change**, **Allow Delete**, or **Allow View** check boxes to enable the appropriate action.

Allow MANY:1 Child Autoinc
Whenever you insert a child record check this box to enable the MultiChildRelationTree to prime the lookup file before calling the child update form. The default is OFF.

OLE control template

This control template helps you implement OLE and OCX controls using the Clarion OLE container control.

For OLE controls the template generates code to activate and deactivate the OLE server and to assign the OLE object to a BLOB for saving.

For OCX controls the template generates appropriate MAP include statements to support the OCX, "shell" callback procedures for the OCX, and code to register the callback procedures. Depending on the OCX used, you may need to add additional functionality to the callback procedures.

See Also:

OCX Library Procedures

OLE Properties

The OLE control template provides the following prompts:

Callback Generation

Callback functions are called by Clarion's runtime library (they are not called directly by your program) when it needs to pass information to your program concerning the OCX. Your callback functions should quickly process the information and respond to the caller. The OLE template can generate shell callback functions and their prototypes so you can concentrate on any special functionality required by the callback.

Event Handler	Check this box to generate an event handling callback function. Clarion's runtime library calls this function when an OCX event needs to be passed to your program.
Property Change	Check this box to generate a property change callback function. Clarion's runtime library calls this function when an OCX property is changed.
Property Edit	Check this box to generate a property edit callback function. Clarion's runtime library calls this function when an OCX property change is requested, but permission to make the change is required from your program before making the change.
Include OCX.CLW in global MAP	Check this box to include prototypes for OCX related functions (callbacks, register callbacks, unregister callbacks, etc.) in your program's global MAP.
Include OCXEVENT.CLW in global data section	Check this box to include common OCX event equates in your program's global data section.
OLE BLOB field	Specify a BLOB to contain the OLE object, or press the ellipsis (...) button to select the BLOB from the *Select Field* dialog.

Pause Button Control template

This control template adds a button to allow pausing and restarting of a Process or Report procedure.

Pause Text	The text to display on the button when the process is running. This text alerts the user that the process can be paused by pressing the button. The default is *Pause.*
Restart Text	The text to display on the button when the process is paused and multiple starts are allowed. This text alerts the user that the process can be restarted by pressing the button. The default is *Restart.*
Start Paused	The state of the control when the procedure starts. If check, the process is paused until the user presses the button.
Start Text	The text to display on the button when the procedure opens. This text alerts the user that the process can be strated by pressing the button. The default is *Go.*
Allow Multiple Starts	Check this button to allow the process to start more than once.
When Pressed	The standard set of prompts for buttons.Normally, when using a Control template, these prompts are not used.

ProcessReportQBEButton

The ProcessReportQBEButton template places a **Query** button on the progress window for a Process or Report procedure. This button allows the end user to apply a dynamic (run-time) filter to a report. The end user can query the underlying data and print the results of the query in a report. This control template requires a PauseButton control template on the Progress Window to allow the end user the opportunity to press the **Query** button.

The prompts for this control template mirror the template prompts for the BrowseQueryButton control template. See *Read-Write Browse Templates—BrowseQueryButton* for the deatiled information on these prompts.

The end user may provide filter criteria for zero or more columns. Additional filter criteria result in a more refined search and a smaller result set (the filter conditions are conjunctive—ANDed together).

Runtime Options

The default comparison operator is (=), which searches for an exact match between the Report field and the corresponding Query input field. By default all matches are case sensitive. Pressing the comparison operator button cycles through all the available operators:

Operator	Filter Effect
=	browsefield equal queryvalue
>=	browsefield greater than or equal queryvalue
<=	browsefield less than or equal queryvalue
<>	browsefield not equal queryvalue
	no filter

For string fields, you may use the following special characters in the Query input field to refine your search:

Symbol	Position	Filter Effect
^	Prefix	caseless (case insensitive) search
*	Prefix	browsefield contains queryvalue
*	Suffix	browsefield begins with queryvalue

For example:

d	- matches 'd' only
d*	- matches 'dog', 'david'
*d	- matches 'dog', 'cod'
^*d	- matches 'dog', 'cod', 'coD'

Upon completion of the Query dialog, the current sort order of the Report is filtered to match the query. If Query is selected again, the previous query is available by default. This allows sharing of filters between sort orders, as well as successive filter refinements.

The standard Query behavior is defined by the ABC Library's QueryClass. See *QueryClass, QueryFormClass,* and *QueryFormVisual* for more information.

The BrowseQueryButton template provides the following prompts:

General

QBE Family
Specify the query family for the QBE interface. Normally this is the name of the procedure.

QBE Interface
Select the query interface from the drop-down list. Choose from:

Form
One input field and button per Query field

List
One listbox row per Query field

Auto Populate
Check this box to provide a query dialog with filter criteria for each field in the Report. The input fields have the same picture token and prompt as the corresponding Report field.

Clear this box to enable the **Fields** button and specify custom query input fields.

Caseless Auto Populate
Check this box to do case insensitive searches for each field in the Report. Clear the box to do case sensitive searches.

Fields
Press this button to populate specific query input fields. You can use this option to restrict the query to some subset of Report fields, or to expand the query to include fields not in the Report. You can also implement caseless searches by default.

Field
Type the field name to include in the Query dialog, or press the ellipsis button to select the field from the **Select Field** dialog.

Title
Type the prompt or label associated with the Query field.

Picture
Type a picture token for the Query field, or press the ellipsis button to select a token with the **Edit Picture** dialog.

Caseless
Check this box to do case insensitive searches on the Query field. Clear the box to do case sensitive searches.

Disable Begins\Contains?	Check this box to disable the Begins\Contains queries. This is available for the QBE form interface for any non-string field.
Retain Query	This option is checked by default, and indicates that the end-user's query will remain in the Query dialog on the subsequent press of the Query button. Clear the check box to reset the Query dialog on each press of the Query button.
Use on startup	Check this box to open the Query dialog before the Browse procedure opens.
Result Control	Optionally select a STRING Control from the Droplist to display the filter statement created by the QBE object. A property assignment is made to the selected control (using PROP:Text), therefore it is not necessary to associate a variable with the STRING.
Auto-share between tabs	Not applicable and disabled always here.
Quick QBE Support?	Not applicable and disabled always here.
Submenu Icon	Not applicable and disabled always here.
Query Icon	Not applicable and disabled always here.

QBE Class

Select this tab to override the global Query Manager setting. See *Template Overview—Classes Tab Options—Global* and *Local*.

QBE Visual Class

Select this tab to override the global Query Manager setting. See *Template Overview—Classes Tab Options—Global* and *Local*.

BrowseQueryButton

The BrowseQueryButton template provides a **Query** button to let the end user apply a dynamic (run-time) filter to the BrowseBox result set. In other words, the end user can query the underlying dataset and display the results of the query in the BrowseBox list.

The default query interface is a dialog with an input field and a comparison operator button for each list box column.

The end user may provide filter criteria for zero or more fields. Additional filter criteria result in a more refined search and a smaller result set (the filter conditions are conjunctive--ANDed together).

Runtime Options

The default comparison operator is (=), which searches for an exact match between the BrowseBox field and the corresponding Query input field. By default all matches are case sensitive. Pressing the comparison operator button cycles through all the available operators:

Operator		**Filter Effect**	
=	*browsefield*	equal	*queryvalue*
>=	*browsefield*	greater than or equal	*queryvalue*
<=	*browsefield*	less than or equal	*queryvalue*
<>	*browsefield*	not equal	*queryvalue*
	no filter		

For string fields, you may use the following special characters in the Query input field to refine your search:

Symbol	**Position**	**Filter Effect**		
^	prefix	caseless (case insensitive) search		
*	prefix	*browsefield*	contains	*queryvalue*
*	suffix	*browsefield*	begins with	*queryvalue*

For example:

d	- matches 'd' only
d*	- matches 'dog', 'david'
*d	- matches 'dog', 'cod'
^*d	- matches 'dog', 'cod', 'coD'

Upon completion of the Query dialog, the current sort order of the BrowseBox is filtered to match the query. If Query is selected again, the previous query is available by default. This allows sharing of filters between sort orders, as well as successive filter refinements.

The standard Query behavior is defined by the ABC Library's QueryClass. See *QueryClass, QueryFormClass, QueryFormVisual, QueryListClass, and QueryListVisual* for more information.

The BrowseQueryButton template provides the following prompts:

General

QBE Family
Enter the family name that this QBE object will use. Normally, the QBE family will default to the procedure name. However, there may be situations where you would like saved queries to be shared between one or more procedures. In this case, use a common family name between these shared procedures.

QBE Interface
Select the query interface from the drop-down list. Choose from

Form One input field and button per Query field

List One listbox row per Query field

Auto Populate
Check this box to provide a query dialog with filter criteria for each field in the BrowseBox. The input fields have the same picture token and prompt as the corresponding BrowseBox field.

Clear this box to enable the **Fields** button and specify custom query input fields.

Caseless Auto Populate
Check this box to provide a query dialog where the filter criteria will not be case sensitive for each field in the BrowseBox.

Fields
Press this button to populate specific query input fields. You can use this option to restrict the query to some subset of BrowseBox fields, or to expand the query to include fields not in the BrowseBox. You can also implement caseless searches by default.

Field
Type the field name to include in the Query dialog, or press the ellipsis button to select the field from the **Select Field** dialog.

Title
Type the prompt or label associated with the Query field.

Picture
Type a picture token for the Query field, or press the ellipsis button to select a token with the **Edit Picture** dialog.

Caseless
Check this box to do case insensitive searches on the Query field. Clear the box to do case sensitive searches.

Disable Begins/Contains?
Check this box to disable Begins/Contains queries. This is available for the QBE form interface for any non-string field.

Retain Query
This option is checked by default, and indicates that the end-user's query will remain in the Query dialog on the subsequent press of the Query button. Clear the check box to reset the Query dialog on each press of the Query button.

Use on startup
Check this box to open the Query dialog before the Browse procedure opens.

Result Control
Optionally select a STRING Control from the Droplist to display the filter statement created by the QBE object. A property assignment is made to the selected control (using PROP:Text), therefore it is not necessary to associate a variable with the STRING.

Auto-share between tabs
Check this box to make the query active to all tabs associated with the browse.

Quick QBE Settings

Check the **Quick QBE Support** to enable special popup menu support for saved queries. A *View* menu item will appear in the Browse box popup menu, with a sub-menu of saved queries. You can customize the *View* menu popup icon by modifying the **Submenu Icon** prompt. Customize the saved query items by modifying the **Query Icon** prompt.

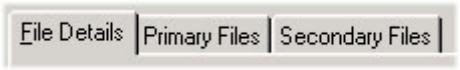
QBE Class

Select this tab to override the global Query Manager setting. See *Template Overview--Classes Tab Options--Global* and *Local*.

QBE Visual Class

Select this tab to override the global Query Manager setting. See *Template Overview--Classes Tab Options--Global* and *Local*.

RelationTree control template



Click on a TAB to see its help

The tree control is a list box formatted to display as a collapsible hierarchical list. This Control template provides an alternative for the Browse-Form paradigm. A single RelationTree control can replace several Browse-Form pairs.

Using the RelationTree Control template, you can specify multiple related files to display on multiple levels (up to 29) of a hierarchical list--*with an associated update procedure for each level*. The related files are declared in the Data / Tables Pad--the Primary (Parent) file and a single chain of related secondary Child files (Parent-Child-GrandChild).

The RelationTree template employs a fully-loaded QUEUE for the root level. The child levels are demand-loaded when a branch is expanded.

Tip

This template is not appropriate for databases with a very large primary file. For large files you should use the BrowseBox Control template.

The plus (+) sign indicates a collapsed level that expands when the user CLICKS on the plus (+) sign. Conversely, the minus (-) sign indicates an expanded level that collapses when the user CLICKS on the minus (-) sign.

To create a tree using the RelationTree Control template:

Place a RelationTree Control template on a window.

This opens the **List Box Formatter**. Use the **List Box Formatter** to enable colorization, icon display, or horizontal scrolling in your tree control (see *The List Box Formatter*). Do *not* use the **List Box Formatter** to populate fields in the tree control.

Tip

The tree control is a single column list, therefore you must specify a column scroll bar rather than a list scroll bar to accomplish horizontal scrolling.

Press the **OK** button on the **List Box Formatter**.

RIGHT-CLICK on the RelationTree Control template and choose **Actions** from the popup menu.

Access the Data / Tables Pad to specify the file schematic for the control.

Specify the Primary (Parent) file and a single chain of related Secondary Child files (Parent-Child-GrandChild).

Complete the RelationTree template prompts.

The RelationTree template provides the following prompts:

File Details

Table Schematic Description

Enter a descriptive string that will be displayed in the Table Schematic window for this particular control. This allows you to distinguish one control from another when there are multiple controls populated in a single window.

Tree Heading Text
An optional text heading at the top of the tree. Tree Heading Text is required to let the user add a record at the root level.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Tree Heading Icon
An optional icon at the top of the tree. Icons must be enabled in the List Box Formatter for this prompt to be enabled.

Expand Branch
Specify a keystroke to expand the selected list item--display its children. Press the ellipsis button (...) to select special keys such as ESC, TAB or ENTER. See *Controls and Their Properties--Common Control Attributes--Setting the KEY Attribute* for more information on this dialog.

Contract Branch
Specify a keystroke to contract the selected list item--hide its children. Press the ellipsis button (...) to select special keys such as ESC, TAB or ENTER. See *Controls and Their Properties--Common Control Attributes--Setting the KEY Attribute* for more information on this dialog.

Accept control from Toolbar
Check this box to accept navigation events and other relation tree control events generated by the *FrameBrowseControl* control template on the APPLICATION's toolbar. See *FrameBrowseControl* for more information on these toolbar buttons and their operation. Clear this box to disable the *FrameBrowseControl* toolbar buttons for this procedure.

Give option to expand and contract all levels
Specify the RIGHT-CLICK popup menu for the RelationTree includes "Expand All" and "Contract All" commands.

Primary Files – Primary File Details

Display String

The field name or text to display for the primary file level. This may be any valid Clarion expression, for example:

```
CLIP(CUST:LastName)&' '&CUST:FirstName
```

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Update Procedure
The update procedure to call for the primary file. The procedure may be accessed with the RIGHT-CLICK popup menu automatically provided when you specify an update procedure. The default popup menu text is "Insert," "Change," and "Delete."

The procedure may also be accessed with the RelationTreeUpdateButtons--see below. If you use the RelationTreeUpdateButtons control template, the popup menu inherits the text from the buttons.

Record Filter
Type a valid Clarion expression to limit the contents of the list to only those records causing the expression to evaluate to true (nonzero or non-blank). The procedure loops through all displayable records to select only those that meet the filter.

You must BIND any file field, variable, or EQUATE that is used in a filter expression. The **Hot Fields** tab lets you BIND fields.

Colors

This tab is only available if you check the **Color Cells** box in the **List Field Properties** in the List Box Formatter.

Default Colors
To specify the default colors for the primary file display string, type color EQUATEs (from \LIBSRC\EQUATES.CLW) in the entry fields or press the ellipsis (...) buttons to select colors from the **Select Color** dialog.

Conditional Color Assignments
To specify conditional colors for the primary file display string, press the **Insert** button. This opens the **Conditional Color Assignments** dialog.

Conditional Color Assignments

This dialog lets you specify the conditional colors for the primary file display string.

Condition
Type a valid Clarion expression to evaluate at runtime, then type color EQUATEs (from \LIBSRC\EQUATES.CLW) in the entry fields or press the ellipsis (...) buttons to select colors from the **Select Color** dialog.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

At run-time these conditions are evaluated, and the colors for the first true condition in the list are used.

Icons

This tab is only available if you check the **Icons** box in the **List Field Properties** in the List Box Formatter.

Default Icon
To specify the default icon for the primary file display string, type the icon filename in the entry field.

Conditional Icon Usage
This dialog lets you specify conditional icons for the primary file display string.

To specify conditional icons for the primary file display string, press the **Insert** button. This opens the **Conditional Icon Usage** dialog.

Press the **E** button to call the Expression Editor. This dialog is used to help you construct syntactically correct expressions to use in the appropriate prompt.

Condition
Type a valid Clarion expression to evaluate at runtime.

Icon
Type the icon filename in the entry field.

At run-time these conditions are evaluated, and the icon for the first true condition in the list is used.

Secondary File Settings

The secondary file settings are identical to the primary file settings. Highlight the secondary file, then press the **Properties** button below the **Secondary Files** list box. See *RelationTree Overview* for information on how to specify the secondary files with the **Select File** dialog.

RelationTree Embed Points

The RelationTree Control template provides a comprehensive set of embed points to allow full customization of the control's behavior.

Relation Tree Update Buttons control template

This Control template adds three buttons (**Insert**, **Change**, and **Delete**) which allow the user to call the associated update procedure for the selected level of a Relation Tree (if an update procedure has been specified) . There are no prompts for this control. The Update Procedure is specified for each level of the Relation Tree control template. .

The Change and Delete buttons correspond to the currently highlighted record. The Insert button adds a child record (the next level down the tree structure).

Relation Tree Expand/Contract Buttons control template

This Control template adds two buttons (**Expand** and **Contract**) which let the user expand or contract all the items in the relation tree. A Relation Tree control template. must be present on the window. There are no prompts for this template.

ReportDateStamp control template

The ReportDateStamp template adds two STRING controls to a REPORT: a "Report Date:" text STRING, and a formatted variable STRING to display the date. By default, the ReportDateStamp template displays the system date using the Windows standard long date format (D18). For example, November 18, 1997. However, you may select an alternative format and an alternative date value to display.

The ReportDateStamp template provides the following prompts:

Format Picture	Press the ellipsis button to select a date format. See *Picture Tokens* in the *Language Reference*.
Use System Clock?	Check this box to display the system date (see *TODAY* in the *Language Reference*). Clear the box to display a variable containing the date value to display.
Date Variable	Type the variable name or press the ellipsis button to select the variable from the **Select Fields** dialog.

ReportPageNumber control template

The ReportPageNumber template adds a variable STRING to display the page number.

The ReportPageNumber template provides no configuration prompts.

ReportTimeStamp control template

The ReportTimeStamp template adds two STRING controls to a REPORT: a "Report Time:" text STRING, and a formatted variable STRING to display the time. By default, the ReportTimeStamp template displays the system time using the Windows standard long time format (T8). For example, 12:90:22 PM. However, you may select an alternative format and an alternative time value to display.

The ReportTimeStamp template provides the following prompts:

Format Picture	Press the ellipsis button to select a time format. See *Picture Tokens* in the *Language Reference*.
Use System Clock?	Check this box to display the system date (see *CLOCK* in the *Language Reference*). Clear the box to display a variable containing the time value to display.
Time Variable	Type the variable name or press the ellipsis button to select the variable from the **Select Fields** dialog.

RTFStatusBar Control Template

This code template requires the population of the RTFTextControl Control Template.

The **RTFStatusBar** Control Template populates a simulated status bar made up of panel and string controls that can be applied to any window.

There are no template prompts for this control. The information contained in this template is shown below:

RTFTextControl Control Template

The RTFTextControl control template provides built-in RichEdit support for native TEXT controls. In previous releases, support for Rich Text Format was provided in a special and detailed set of special support templates. This new RTFTextControl allows for easy integration of RTF support into all of your applications, with additional code and control template support through the RTFToolbar, RTFStatusBar, and the RTFAction Code Template.

Note:

This template (and its associated support templates) replaces the RTF Control template, and its associated extension and code template support.

In essence, this template simply adds a TEXT control to your window, with the RTF attribute applied. It also uses a *Field* as the RTF **Value Mode** parameter.

Tip

The normal design flow of the new RTF text control support is as follows:

1. Populate the RTFTextControl
2. Populate the RTFToolbar
3. Populate the StatusBar
4. Add the RTFAction Code template to additional control and event embed points if needed.

Context Menu

The Context Menu interface allows you to construct custom menu items to add to the existing popup menu used with the RTF control. You can select from a list of actions to attach to each menu item that you create.

Press the appropriate button to add (**Insert**), modify (**Properties**), or remove (**Delete**) a context menu item. The following options are available or the subsequent dialog.

Is a separator?
Check this box if you would like to add a separator line to your context menu. There are no other prompts associated with this action.

Menu Text
Enter the name of the context menu item to display.

When Pressed

No Special Action
The menu item will have no template-generated action associated with it. This option normally is used when you are using an embed point to control this menu's action.

Execute Routine
Enter a valid **Routine Name** here. Of course, you must have the ROUTINE defined within the scope of this procedure.

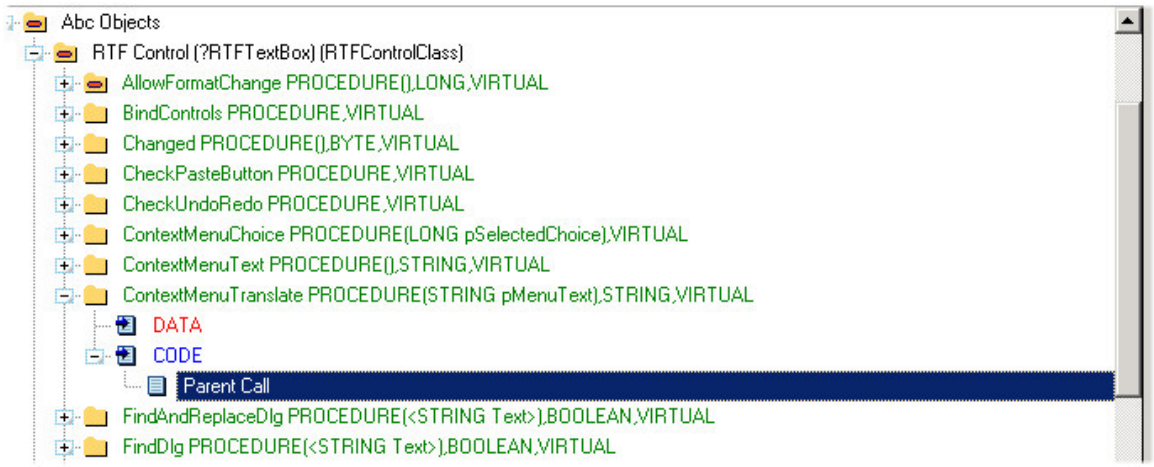
Post Event to Control
Select a valid Field Equate from the drop list to post an event to the **Control.** Select an *Accepted* or *Selected* **Event** to post to the control from the drop list provided.

Call a Procedure
The RTF Context Menu interface uses the standard interface for calling a procedure. If you need more detailed help here, see the following topic.

Run a Program
The RTF Context Menu interface uses the standard interface for running a program. If you need more detailed help here, see the following topic.

Tip

You can translate the contents of the default popup menu items by using the *ContextMenuTranslate* method *Parent Call* embed point as shown below:



The exact texts of standard (default) items of the standard RTF control context menu are as follows:

&Undo

Copy

Cut

&Paste

Delete

Select &all

Insert object...

Paste special...

Example code added to *ContextMenuTranslate* Parent Call:

```
!This example only translates three menu items
  CASE CLIP (pMenuText)
  OF 'Cut'
    RETURN 'Move'
  OF 'Copy'
    RETURN 'Duplicate'
  OF '&Paste'
    RETURN 'Paste Clipboard'
  ELSE
    RETURN ''
  END
```

Classes

The Classes tab lets you control the classes (and objects) your procedure uses to accomplish the template's task—that is, they override the global class settings specified in the **Global Properties** dialog. You may accept the default Application Builder Class specified in the **Global Properties** dialog (recommended), or you may specify your own or a third party class to override the default setting. Deriving your own class can give you very fine control over the procedure when the standard Application Builder Class is not precisely what you need.

Object Name
Set the object's label for the template-generated code.

Use Default Application Builder Class?
Check this box to use the default Application Builder Class specified in the **Global Properties** dialog. Clear this box to use a class other than the default, and to enable the following prompts.

Use Application Builder Class?
Check this box to select a class from the **Base Class** drop-down list. The list includes all classes with the LINK attribute in \LIBSRC*.INC files. Clear this box to specify a class declared elsewhere.

Base Class
If you checked the **Use Application Builder Class?** box, select a class from the drop-down list. If you cleared the **Use Application Builder Class?** box, type the class label here, and type the name of the source file that contains the class declaration in the **Include File** entry box.

Include File
If you cleared the **Use Application Builder Class?** box, type the class label in the **Base Class** entry box, and type the name of the source file that contains the class declaration here.

Derive?
Check this box to derive a class based on the parent class specified above and to enable the **New Class Methods** and **New Class Properties** buttons to define any *new* properties and methods for the derived class.

This prompt is primarily to allow you to define *new* properties and methods in a derived class. To override *existing* methods, simply embed code in the corresponding method embed points.

Using **Derive?**, **New Class Methods** and **New Class Properties** makes the template generate code similar to the following:

```
MyProcess CLASS(Process) !derive a class from the parent class
NewMethod PROCEDURE !prototype new class method
NewProperty BYTE !declare new class property
END
```

The template automatically derives from the parent class if you embed code into any of the derived method embed points, regardless of the status of this check box.

New Class Methods
Press this button to specify the *new* method prototypes to generate into the derived CLASS structure. This opens the **New Class Methods** dialog (see *New Class Methods*).

New Class Properties
Press this button to specify the new property declarations to generate into the derived CLASS structure. This opens the **New Class Properties** dialog (see *New Class Properties*).

Application Builder Class Viewer
Press this button to display classes, properties, and methods used by the ABC Templates, and the relationships between parent and derived (child) classes. This utility can help you analyze and understand the classes that the ABC Templates use.

Refresh Application Builder Class Information
Press this button if you have changed the contents of an include file (.INC) or added an include file to the \LIBSRC directory. Typically, this is needed when you install third party products that use ABC compliant classes, although you may create your own ABC compliant classes too. See *ABC Compliant Classes* for more information. The ABC Templates use information gleaned from the header files for generating embed points, loading the Application Builder Class Viewer, application conversion, etc.

Composite Class
Press these buttons to open a Classes dialog for each class used by the parent class specified above. For example, the WindowManager uses a Toolbar class, so the WindowManager's Classes dialog contains a Toolbar Class button to open a Classes dialog for its Toolbar Class.

RTFToolBar Control Template

This control template requires the population of the RTFTextControl Control Template.

The **RTFToolBar** Control Template populates a group of support controls to use with the RTFTextControl control template.

There are no template prompts for this control. You can disable or hide controls that you do not want to provide to the user. You can also modify the icons and text used on each support control.

Each control is described as follows:

 **New**	Clears the contents of the RTF Control.
 **Open**	Prompts you to load an external RTF file's contents into the RTF control
 **Save**	Allows you to save the existing RTF control's contents to an external file.
 **Print**	Prints the RTF control contents.
 **Find**	Opens the *Find* dialog window, allowing you to search for text in the RTF control.
 **Find and Replace**	Opens the *Find and Replace* dialog window, allowing you to search and replace target text in the RTF control.
 **Cut, Copy and Paste**	Performs the standard Windows cut, copy and paste actions for selected text in the RTF control.
 **Undo and Redo**	Allows you to *undo* the sequence of actions performed on an RTF control, and to *redo* those actions if needed.
 **Tab Stops**	Press this button to access the *Tabs* dialog. Set your tab stops in millimeters, inches, or twips. The hotkey to move to the next tab stop is **CTRL+TAB**
 **Paragraph Indents**	Press this button to access the *Paragraph Indents* dialog. Set the **First** line, **Left** side and **Right** side of the paragraph in millimeters, inches, or twips.
 **Alignment Controls**	These buttons provide the standard alignment features (left, right, center, and full justification) for the selected text.

Note:

Full justification is currently only supported in Rich Edit versions greater than 3.0. The presence of RichEdxx.DLL is required in the Windows *System* directory.

Bullets and Numbering/Bullet Styles

Add bullets or numbers and set the style with these buttons. Bullets and numbering can only be applied at the *start* of a paragraph.

Font and Colors

These controls set all aspects of the active font and color to use in the RTF control.

Hiding or Removing Controls

If there are controls that you do not need in your application (Example: *Save* or *Print*), you can simply delete them. When you are prompted with "Do you want to delete this control template?" Press **No**, and only the individual control will be removed.

To conditionally hide controls at runtime, you can use the following embedded source in the Window Manager's Init Method (just *after* the first RTFControl Initialization embed point):

```
IF NeedToHideSave          !locally defined flag
    HIDE(?RTFToolSave)     !hide Save button
    RTFControl19.AddItem(RTFToolbar:CtlButtonSave,0)
    !the AddItem method disables Save button logic
END
```

Use this technique for any button that you need to conditionally display.

Save Button control template

The SaveButton template provides an **OK** button for your window, plus the capability to display an action message for the end user. The SaveButton handles most of the file I/O for the procedure.

The SaveButton template provides the following prompts:

Allow
Check any combination of the three boxes to specify permitted file I/O operations. Conversely, clear the box to prevent the associated operation.

Inserts	Generates code to handle record inserts.
Changes	Generates code to handle record changes.
Deletes	Generates code to handle record deletes.

The SaveButton template does not detect changes to BLOBs; therefore, if only the BLOB changes, the SaveButton template does not save it.

Field Priming on Insert
Field Priming lets you provide a default value for fields in a new record. This value supersedes any initial value specified in the data dictionary. You can select a field and set an initial value in the **Field Priming** dialog.

Messages and Titles
Press this button to open the **Messages and Titles** dialog to specify update messages and their locations. In addition, this dialog controls some fundamental behavior associated with the procedure, such as whether it confirms before canceling and whether it allows repetitive adds.

View Message
Specifies the text for the action message when the procedure is called to view a record.

Insert Message
Specifies the text for the action message when the procedure is called to add a record.

Change Message
Specifies the text for the action message when the procedure is called to change a record.

Delete Message
Specifies the text for the action message when the procedure is called to delete a record.

On Aborted Add/Change
Specifies the action to take when the user presses the **Cancel** button while adding or modifying a record. Choose from:

Offer to save changes
Displays a message box prompting to save changes before cancelling.

Confirm Cancel
Displays a message box prompting asking if you really want to cancel.

Cancel without Confirming
Displays no message before cancelling.

Field History Key
Specify a key that restores the value from the last saved record. When the end user presses the specified key, the generated code retores the field with focus from the previously processed record.

The default key is **CTRL + H**.

Specifying a key here also enables the FrameBrowseControl's "ditto" button  . This button also restores the value from the last saved record.

When called for Delete
Specify what displays when this procedure is called to delete a record.

Choose from:

Standard Warning
Displays a message box prompting for confirmation of the delete.

Show Form
Displays the form.

Automatic Delete
Allows records to be deleted without a display or prompt for confirmation.

After successful insert
Select one-at-a-time insert mode or repetitive insert mode.

Choose from:

Return to caller
Generates a RETURN to the calling procedure following a successful insert. This results in a one-at-a-time insert mode.

Insert another record
Does not generate a RETURN to the calling procedure following a successful insert. This results in a repetitive insert mode.

Ask the user before adding another record
Does not automatically generate a RETURN to the calling procedure following a successful insert, but asks the user whether to add another record.

Location of Message
Specifies where the message displays.

Choose from:

None/Window Control
Embed your own code to display the message in a control.

Title Bar
Display the message in the window's title bar.

Status Bar
Display the message in the window's status bar. Optionally specify which section of the status bar in the **Status Bar Section** box.

Display Record Identifier on the Title Bar
Check this box to append a string to the caption on the window's titlebar. Specify the string in the **Record Identifier** field.

Record Identifier
Specifies the string to append to the titlebar caption, which you can use to identify the record. Type a string in the Record Identifier box. To use a variable name, precede it with an exclamation point (!).

SaveChangeButton Control Template

The SaveChangeButton template provides an icon "Save" button  for your form window (or any window that has a Save Button template populated). The purpose of this template is to allow a user to update a form as they navigate from field to field. This can be particularly useful with very large form dialogs.

The SaveButton template provides the following prompt:

Disable Cancel button after Save
Check this box to disable the Cancel button after the user has pressed the SaveChange button. The logical exit from the form after a save is through the OK button (Save Button) only.

Sort Order Button control template

The Sort Order Button control template is designed to provide a button control that is the central sort order control of a target browse box. When populated, the sort order button automatically hides the tab controls (and optionally, the sheet control), and when pressed, displays a pop up menu of sort order selections.

Sheet Control	Select the Sheet control that contains the tab controls that control the browse box sort orders. Each tab control represents a different browse sorting sequence. The default sheet control Field Equate Label is *?CurrentTab*, but you can select any valid sheet control in the drop list provided.
Hide Sheet	Check this box to hide the sheet (and associated tab controls) that you selected from the **Sheet Control** prompt.

Sort Order Drop List control template

The Sort Order Drop List control template is designed to provide a Drop List control that is the central sort order control of a target browse box. At runtime, the tab controls (and optionally, the sheet control) are hidden, and the drop list displays a list of sort order selections.

As you initially populate the Drop List control, press **Cancel** when the *Field Selection Dialog* is displayed. Press the **No** Button when you are asked to **Save Properties** in the List Box Formatter. The template will automatically populate the drop list box with sort order choices. The only required prompts are described below.

Another tip; when you populate the drop list, make sure that it is not placed on a tab control, but rather, on the window itself. This will make it visible for all sort choices. You can verify this in the Window Designer's Property Editor.

Sheet Control	Select the Sheet control that contains the tab controls that control the browse box sort orders. Each tab control represents a different browse sorting sequence. The default sheet control Field Equate Label is *?CurrentTab*, but you can select any valid sheet control in the drop list provided.
Hide Sheet	Check this box to hide the sheet (and associated tab controls) that you selected from the **Sheet Control** prompt.

Procedure Templates

Select Procedure Template or Defaults

Wizards

Template Wizards are the most powerful design tool within the Application Generator. Here, you can select a base procedure (Browse, Form, Report, etc.) or a whole application, and then fine tune the wizard prompts and options to produce a near perfect application or procedure that fits your specifications.

See Template Wizards for more detailed information.

Defaults

The Defaults tab allows you to select from a wide variety of pre-defined structures and functionality. Based on the type of default you select, the template procedure associated with it is automatically attached upon your selection.

Templates

This dialog lets you choose a procedure template, adding functionality to any new or "To Do" procedure in the Application Tree .

CLICK on a procedure template from the list, then press the Select button. Once you select a procedure type, you can customize it using its Procedure Properties dialog.

If you add third party, or your own customized templates to the Template Registry, they appear in the list. The following lists the procedure templates which ship with Clarion:

Browse	Browse fields in a page-loaded list box.
External	A procedure contained in an external library (*.LIB only) or object file
Form	View/edit a single record from file
ViewOnlyForm	View a record from a file
Frame	Multiple Document Interface (MDI) main menu
Menu	Single Document Interface (SDI) menu
Process	Sequential record (batch) processor
Report	Generic reporting procedure
Source	Source procedure--add hand coded source to your application
Splash	Display a splash screen
Viewer	View an ASCII text file

Window	Generic window handler

ADO Browse Procedure

The ADO Browse Procedure is used to display a list of records in a standard list box, using an ADO connection layer to retrieve the data elements (or recordset). The default procedure is a combination of the ADO Browse Box control, ADO BrowseUpdate control, and a standard Close Button control. The ADO Browse procedure is supported for both ABC and Clarion template chains.

See Also:

SoftVelocity ADO Browse Box

ADO Browse Update Buttons

ADO Form

The ADO Form procedure includes the template elements necessary to allow updates of an active ADO connection.

See Also:

ADO Save Button Control

ADO Login Procedure Template

The ADO Login Procedure Template provides a quick and easy way to allow ADO connections using a variety of different techniques.

Two templates are associated with this procedure.

ADO Process Procedure

The ADO Process Procedure contains the template elements necessary to call a process that affects an existing ADO connection. An optional Pause ADO Process Button can also be added to the existing defaults.

See Also:

ADO Process Control Extension

ADO Pause Process Button Control

ADO Report Procedure

The ADO Report Procedure contains the template elements necessary to call a process that returns, and prints, a recordset of an existing ADO connection. The ADO Report Procedure is nearly identical to the ADO Process Procedure template, with the addition of a default report structure, and preview/print logic. An optional Pause ADO Process Button can also be added to the existing defaults.

See Also:

ADO Process Control Extension

ADO Pause Process Button Control

Wizard and Utility Templates

Select Utility Dialog

A Utility template lets you produce output from your application. These templates can provide extensible supplemental utilities for such things as wizards, program documentation, or a tree diagram of procedure calls.

Highlight the desired utility template, then press the **Select** button.

Clarion provides *WIZARDS*--powerful utility templates that let you create a Browse, Form, or Report procedure by answering a few quick questions. You can even use a wizard to create an entire Application from an existing dictionary!

Options you specify in advance in the Data Dictionary provide additional control over the procedures the wizards create. See **Using Wizard Options** for more information.

Application Wizard	Creates a complete application from an existing dictionary.
Browse Wizard	Creates a multi-keyed browse procedure from an existing dictionary table.
Dictionary Print Wizard	Prints information from your data dictionary, from full detail to high level summary.
Form Wizard	Creates an update procedure from an existing dictionary table.
Label Wizard	Creates a label style report procedure from an existing dictionary table.
Process Wizard	Creates a process procedure from an existing dictionary table.
Quick Start Wizard	Creates a Data Dictionary and an application based on the dictionary.
Report Wizard	Creates multi-keyed report procedures from an existing dictionary table.
Theme Maintenance Wizard	Allows you to create and maintain wizard themes, which are used as design criteria for the look and functions of the template wizards.
Window Wizard	Creates a generic window procedure.

Application Wizard utility template

This wizard creates a complete application from an existing dictionary. It creates a Frame containing a menu with options calling all procedures it creates. It also creates Browse and Report procedures for each specified file, with associated Form (Update) procedures.

Answer the questions in each dialog, then press the **Next** button.

After the introduction screen, you are presented with the following options:

Theme Selection:

Theme
Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.

Report Layout
Select a default report layout from the drop list provided. This layout will be the basis for all of the reports that will be generated by the wizard.

If you are using the **Quick Start Wizard**, this is the only dialog that you will be presented with. Press the Finish button to begin the creation of your application.

Other prompts that follow on subsequent windows:

Generate Procedures for:

all files in my dictionary
Select this item to instruct the wizard to generate browse/form and optional report procedures for all tables defined in the dictionary.

primary files
Select this item to instruct the wizard to generate browse/form and optional reports for all tables defined in the dictionary that are not defined as relational child tables to any other table. This option is useful for applications with large dictionaries whose relationships have not yet been defined, and limits the generation of additional procedures used to establish these related tables (child browse procedures and selects).

selected files
Select this item to instruct the wizard to generate browse/form and optional report procedures for all tables selected in the next dialog window, which is a list of all tables defined in the selected dictionary.

For SQL based file systems, the Application Wizard also generates code to capture user login information upon initial program load, and then reuse the login information for each file accessed.

Which control model should the Application use?

There are three models the wizard can use to create applications: Button, Toolbar, or Both. See Application Wizard-Control model .

Button
The wizard builds the application with traditional Insert, Change, Delete, OK, and Cancel command buttons that appear on each dialog.

Toolbar
The wizard builds the application with global toolbar command buttons that appear on the application frame. The toolbar buttons control each dialog. See Control Templates-FrameBrowseControl for more information.

Both
The wizard builds the application with both the traditional dialog command buttons and the global toolbar command buttons.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press one of the customization buttons to modify them at this time.

Frame CustomizationTPLWizardFrameCustomization

Browse CustomizationTPLWizardBrowseCustomization

Form CustomizationTPLWizardFormCustomization

Report CustomizationTPLWizardReportCustomization

Create an Internet Enabled Application

If you are using the ABC template chain, check this box to apply the Web Application Extension (WebBuilder) templates to your application. If you are using the Clarion template chain, check this box to apply the Internet Application Extension (Internet Connect) templates to your application. In both instances, this allows your application to be deployed as both a Windows and Internet application.

You must have the appropriate template set registered in order to use this feature.

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

Generate Reports for each file

Check this box to automatically generate report procedures. Clear the box to omit report procedures.

Select Sort Order

Select from the drop list the sort and report generation method from the following choices:

Single Key

Select this option to force the wizard to generate a separate report for each key defined in your file (or files).

Runtime Key Selection

Select this option to force the wizard to generate a single report that pops up a sort order dialog prior to printing at runtime.

Record Order

Select this option to force the wizard to generate a single report sorted by record order for your selected file (or files).

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save on a new theme,** and enter the new name of the Theme and Theme file**.**

You can press the **Back** button to change a prior selection or press the **Cancel** button to abandon the application.

The Application Wizard creates the .APP file based on the dictionary and the answers you provided, and then displays the Application Tree dialog for your new application.

Fine Tuning the Wizard

You can control how the wizard builds your application by specifying options for Tables, Columns, Keys, and Relationships in the Data Dictionary (see Dictionary Options Dialog).

Wizard--Control model

The Application Wizard creates one of three application types based on the model you specify.

Button The wizard builds the application with traditional **Insert, Change,** and **Delete** command buttons that appear on each dialog.

Toolbar The wizard builds the application with global toolbar command buttons that appear on the application frame.

Both The wizard builds the application with both global toolbar command buttons that appear on the application frame and traditional **Insert, Change,** and **Delete** command buttons that appear on each dialog.

Browse Procedure Wizard utility template

This wizard creates a multi-keyed Browse Procedure from an existing dictionary file definition. It also creates associated Form (Update) procedures, if you specify that updates be allowed.

There are three models the wizard can use to create procedures: Button, Toolbar, or Both. See Wizard-Control model

After the introduction screen, you are presented with the following options:

Theme Selection

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Save Settings	After you have selected a theme, you have the option to save these settings for any future applications that you create.

What name should be used as the label of the procedure?

Type the browse procedure name.

Which file do you want to browse?

Press the ellipsis (...) button to select a file from the dictionary.

Browse using all record keys

Check this box to make the list sortable on all keys. Clear the box to specify a single sort key.

Allow the user to update records

Check this box to generate a subordinate procedure to update the table. Optionally, provide the name of the update procedure. Clear the box to make the list read only.

Call update using popup menu

Check this box to provide right-click popup menus on the Browse list in addition to any command or toolbar buttons.

Parent Record Selection

This prompt appears only if you specify a single sort key that is the linking key in a Many:One relationship. The Browse Wizard infers from this that you may want to browse only the child records for a specific parent record. Select one of the following to confirm or deny this inference.

Do not select by parent record

Do not limit the browse - in other words, browse all records.

Select parent record via button

Browse only the child records for a specific parent record. Provide a button to select the parent record.

Assume that the parent record is active

Browse only the child records for a specific parent record. Assume the parent record is already active.

Provide buttons for child files

Check this box to provide buttons on the Browse window to access related child tables. Alternatively, related tables may be accessed from the generated update procedure.

Provide a "Select" button

Check this box to provide a "Select" button that displays when the Browse procedure is called to select a record, but is hidden when the Browse is called to update records.

Which control model should the Application use?

Button	The wizard builds the browse with traditional Insert, Change, and Delete command buttons that appear on each dialog.
Toolbar	The wizard builds the browse to use global toolbar command buttons that appear on the application frame. See Control Templates - FrameBrowseControl.
Both	The wizard builds the browse to use both traditional dialog command buttons and global toolbar command buttons.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press one of the customization buttons to modify them at this time.

Browse CustomizationTPLWizardBrowseCustomization

Form CustomizationTPLWizardFormCustomization

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save to a new theme.**

The Browse Procedure Wizard creates the procedure(s) based on the dictionary file and the answers you provided, and then displays the Procedure Properties dialog for your new procedure.

Form Wizard utility template

This wizard creates an update Form Procedure from an existing dictionary file definition. The form displays and updates a single record.

After the introduction screen, you are presented with the following options:

Theme Selection:

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Save Settings	After you have selected a theme, you have the option to save these settings for any future applications that you create.

What name should be used as the label of the form procedure?

Type the procedure name.

Which file do you want the form to update?

Press the ellipsis (...) button to select a file from the dictionary.

Allow Records To Be Added

Check this box to allow new records.

Allow Records To Be Modified

Check this box to allow records to be changed.

Allow Records To Be Deleted

Check this box to allow records to be deleted.

Insert Message

Type the title bar text to display when adding a record.

Change Message

Type the text to display when changing a record.

Delete Message

Type the text to display when deleting a record.

Where do you want this message to be displayed?

Choose the title bar or the status bar.

A field can be displayed that identifies the active record.

Press the ellipsis button to select a column from the dictionary to display on the window title bar.

Validate field values whenever field value changes?

Check this box for immediate validation when the end user "accepts" the column.

Validate field values when the OK button is pressed?

Check this box for column validation on the OK button.

Browsing child files

Select one of the following choices.

Place children on tabs

Access children with push buttons

Do not provide child access

Which control model should the Application use?

Button	The wizard builds the dialogs with traditional Insert, Change, and Delete command buttons.
Toolbar	The wizard builds the form to use global toolbar command buttons that appear on the application frame. See Control Templates - FrameBrowseControl.
Both	The wizard builds the form to use both traditional dialog command buttons and global toolbar command buttons.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press one of the customization buttons to modify them at this time.

Form CustomizationTPLWizardFormCustomization

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save to a new theme.**

The Form Procedure Wizard creates the procedure(s) based on the dictionary table and the answers you provided, and then displays the Procedure Properties dialog for your new procedure.

Label Wizard utility template

This wizard creates a report procedure from an existing dictionary file definition that includes a defined report label layout.

After the introduction screen, you are presented with the following options:

Theme Selection:

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Label Group	Select a label group from the drop list. A label group contains the most popular classes of labels (Avery, Card Products, InkJet, etc.). If your type of label is not listed here, select **Others**.
Label Type	Select a label type from the drop list. A label type normally corresponds to its product code.
Save Settings	After you have selected a theme, you have the option to save these settings for any future applications that you create.

What name should be used as the label of the report procedure?

Type the procedure name.

Which file do you want to report?

Press the ellipsis (...) button to select a file from the dictionary.

A report can use a single record key, or can run in record order. Enter a key below, or leave the field blank to run in record order.

Press the ellipsis (...) button to select a sort key. Leave the field blank to specify no sort key.

Select the fields that you want the report to use?

Build your report in this list box by adding and deleting fields from the selected file. You can also modify the properties of the fields regarding column labels, picture tokens, and justification. Use the arrow buttons to specify the order that each field will appear on the report.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press the customization button to modify it at this time.

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save to a new theme.**

ProcedureCallTreeReport Utility Template

The ProcedureCallTreeReport Utility Template outputs the Application Generator's Procedure Call Tree display to a printed or text file output.

To use this template:

1. Open an application that uses the dictionary.
2. Choose **Application ▸Template Utility** from the menu, or press **CTRL + U**. The Select Utility dialog appears.
3. Highlight *ProcedureCallTreeReport* and press the **Select** button. The Procedure Call Tree Print Wizard dialogs appear.
4. Answer the question(s) in each dialog, then press the **Next** button.
5. After the first dialog, the Finish button is enabled. Press the **Finish** button now to print all the information available for the Application Tree.

Process Wizard utility template

This wizard creates a process procedure from an existing dictionary file definition.

After the introduction screen, you are presented with the following options:

Theme Selection:

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Save Settings	After you have selected a theme, you have the option to save these settings for any future applications that you create.

What name should be used as the label of the Process procedure?

Type the procedure name.

Which file do you want to process?

Press the ellipsis (...) button to select a file from the dictionary.

A process can use a single record key, or can run in record order. Enter a key below, or leave the field blank to run in record order.

Press the ellipsis (...) button to select a sort key. Leave the field blank to specify no sort key.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the Next button to accept the selected theme's settings, or press one of the customization buttons to modify them at this time.

Process CustomizationTPLWizardProcessCustomization

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save to a new theme.**

Report Wizard utility template

This wizard creates a Report Procedure from an existing dictionary file definition.

After the introduction screen, you are presented with the following options:

Theme Selection:

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Report Layout	Select a default report layout from the drop list provided. This layout will be the basis for all of the reports that will be generated by the wizard.
Save Changes?	After you have selected a theme, you have the option to automatically save changes to this theme for any future applications that you create. Check this box to store changes when completing the wizard.

What name should be used as the label of the report procedure?

Type the procedure name.

Which file do you want to report?

Press the ellipsis (...) button to select a file from the dictionary.

Key Sequence - Select Sort Order

Select from the drop list the sort and report generation method from the following choices:

Single Key

Select this option to force the wizard to generate a seperate report for the key that you select in the Enter a key prompt that follows.

Runtime Key Selection

Select this option to force the wizard to generate a single report that pops up a sort order dialog prior to printing at runtime.

Record Order

Select this option to force the wizard to generate a single report sorted by record order for your selected file (or files).

How many columns do you want the report to use?

Type the number of columns for your report. The Report Wizard distributes the report columns evenly across the columns.

Select the fields that you want to use

Build your report in this list box by adding and deleting fields from the selected file. You can also modify the properties of the fields regarding column labels, picture tokens, and justification. Use the arrow buttons to specify the order that each field will appear on the report.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press the **Report Customization** button to modify it at this time.

Report CustomizationTPLWizardReportCustomization

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the Finish button is enabled. If you are satisfied with your answers, press the Finish button. You also have the option here to Save Changes, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to Save to a new theme.

The Report Procedure Wizard creates the procedure based on the dictionary table and the answers you provided, and then displays the Procedure Properties dialog for your new procedure.

Window Wizard utility template

This wizard creates a window using template themes and other basic settings.

After the introduction screen, you are presented with the following options:

Theme Selection

Theme	Select from the drop list of themes. Themes are groups of settings that control colors, fonts, icons, backgrounds, positions and much more - for Frame, Browse, Form and Report procedures. You will have the opportunity to create a new theme as you progress through the wizard. Select a starting or default theme here.
Save Settings	After you have selected a theme, you have the option to save these settings for any future applications that you create.

What name should be used as the label of the procedure?

Type the window procedure name.

Customization

Wizards have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Press the **Next** button to accept the selected theme's settings, or press the Window Customization button to modify it at this time.

Overwrite existing procedures

Check this box to overwrite existing procedures with the same names. Clear the box to preserve existing procedures.

On the last dialog, the **Finish** button is enabled. If you are satisfied with your answers, press the **Finish** button. You also have the option here to **Save Changes**, where any changes to customization options are saved to the theme that you selected at the start of the wizard. If you wish, you can opt to **Save to a new theme.**

Write Help IDs (WriteHLPIDs) Utility Template

The WriteHLPIDs utility template searches through an active application and extracts all help IDs specified by any existing HLP attribute on a WINDOW or control.

If a WINDOW HLP ID is found, the procedure name is extracted and exported with the Help ID to a designated text file. The Field Equate Label is also extracted for controls that have a Help ID assigned.

The following prompts are provided:

Output File

Press the ellipsis button to select a file to generate output to, or enter a new name in the entry field provided.

Skip Items with no HLP attribute

Check this box to only generate procedures and controls that have an HLP attribute attached to them. This could be very convenient for large applications.

Example Text Output:

```
********* List of All HLP IDs for appdemoa *********

Application Help File: myhelp.hlp
   - HTML Help is Enabled for this Application
     Default Help File Name: cwhh.chm
     Append .HTM to Help IDs is ON

  WALink                !procedure name
    ?sSearch:Prompt     !control feq
    ?sSearch
    ?ButtonLookup
    ?String1

  wKeyword  -  'testid.htm'   !procedure with HLP attribute
    ?sSearch:Prompt
    ?sSearch  -  'test2.htm'  !control with HLP attribute
    ?ButtonLookup
    ?String1
```

Note:

The leading tilde (~) is always stripped out of the generated output. The .HTM extension is visible when set by the Global HTML Help Template.

Dictionary Print Wizard utility template

This wizard prints a description of an existing dictionary file at varying levels of detail for files, fields, keys, and relationships. You may print to the printer or to a file.

To use the Dictionary Print Wizard:

1. Open an application that uses the dictionary.
2. Press the **Template Utility** button  located in the Application toolbar menu. The **Select Utility** dialog appears.
3. Highlight **DictionaryPrint** and press the **Select** button.

The Dictionary Print Wizard dialogs appear.

4. Answer the question(s) in each dialog, then press the **Next** button. After the first dialog, the **Finish** button is enabled.
5. Press the **Finish** button now to print all the information available for all the files, fields, keys, and relationships.Or, step through the wizard's dialogs, to select specific files, plus the level of detail to print (All, Some, or None) for the various dictionary components.

Wizard Themes

The Application Generator is packed with special template-based *wizards*, which are designed to help you generate rich, full-featured applications or individual procedures, based on a series of simple information prompts and options.

These wizards also have different "look and feel" settings and actions called *themes*, which can be modified and saved for use in other applications. Themes are set and controlled by a variety of customization options.

Browse Customization

The general appearance of the Browse Wizard is controlled by the settings on this window. These settings can be saved in a "theme" for use in future applications.

Procedure Name
Enter a name that the Browse Wizard will use to generate procedure names in your application. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for each file selected by the wizard. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names.

Browse Message
Enter default text that the Browse Wizard will generate in the MSG attribute of each list box. The user will see this text in the status bar of the runtime application when the list box is selected. The %FileName macro is not required here, but recommended.

Window
Caption
Text that is entered on this line will be used as the description of the window used by the Browse Wizard.

Secondary Caption
Text that is entered on this line will be used as the message that appears in the status bar of a window only when the browse box is populated as a secondary file on a Form.

Images:
Background
Enter a default image here to use as a graphic or watermark for your Browse window.

Mode
If you have designated a background image to use for your Browse window, this option becomes available to control if the image is tiled, stretched, or centered.

Icon
Enter a default icon to use for your Browse procedure’s window. This will allow your Browse window to be minimized if needed.

Font
Press the Font button to select a default font to use for the Browse procedure. Sample text shown below the button is provided to allow you to review your selection.

Options
Select from the drop list to designate the initial position of your Browse window. You can center the window, or use the default position that is set by the template wizard.

In addition, click on the **System Menu** check box to add the Windows System Menu to your Browse procedure. If you will be using any entry fields on the Browse window, you can also click on the **Entry Patterns** check box to allow special formatting picture information (Example: phone numbers or date pictures)

Tabs
Tab Text
Enter text that will be used for each tab control that is generated by the Browse wizard. The default setting is ***&%#) %Key.*** The ampersand (&) identifies the next character as the hot key of the tab control. The **%#** macro identifies the instance or order of the key
used in the tab control. The %Key is the description of the key name (or label of the key if the description in the Dictionary Editor is blank).
Example: *2) By Account*

Sort Order Selection Style
Press the **Select Style** button to access the Select Style dialog window.

Buttons
The Buttons tab is the central control for all buttons used on the window. There is a default **Width** and **Height** setting (in dialog units) used for all buttons. However, each button can be overridden individually through the Button Customization dialog.

The default buttons of the Browse wizard are:
Insert
Change
Delete
View
Select
Close
Parent Select
Help

The generation of each button is controlled by other settings within the template wizard. However, you can also override the generation of the **View** and **Help** buttons by unchecking the appropriate boxes.

Button Customization

This window contains the generic settings used for selected buttons generated by the wizards. The title of the window reflects which button's properties is currently being modified (i.e., Insert Button, Close Button, etc.).

Common properties include:

Procedure Name (Only valid for the Parent Select button)

Enter a procedure name that the button will call to select a parent record. This button is normally displayed in Browse procedures that use a Child file as the primary file. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for the parent file that is related to the child procedure. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names.

Child Browse Proc Name (Only valid for the Child button)

Enter a procedure name that the button will call to browse a child file related to the active primary file. This button is normally displayed in Form procedures to browse a Child file's records to view or modify. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for the child file that is related to the Form's parent. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names.

Window Name (Valid for the Parent Select and Child Buttons)
Text that is entered on this line will be used as the description of the window used by the procedure that is called by the button.

Position
Choose from the drop list an approximate location to place this button, or select **Other** from the drop list to set custom position parameters.

X
Enter an integer constant that specifies the horizontal position of the top left corner of the control.

Y
Enter an integer constant that specifies the vertical position of the top left corner of the control.

Width
Enter an integer constant that specifies the width of the specified control

Height
Enter an integer constant that specifies the height of the specified control.

Text
Enter the text to display on the button

Icon
Enter an icon file name, or use the ellipsis button to select an icon file that you wish to use for this button.

Cursor
Enter a cursor file name, or use the ellipsis button to select a cursor file that you wish to use for this button. The cursor will be displayed when the mouse moves over the button area.

HotKey
Enter a Clarion keycode or equate, or press the ellipsis button to select a key to use as the hot key for this button. Pressing the hot key at program runtime should activate the button.

Message
Enter a message to display in the status bar by default when the mouse moves over the button area.

Tip
Enter a tool tip message to display when the mouse moves over the button area.

Options
Activate the **Flat** check box to give the selected button a flat appearance.

Activate the **Skip** check box to disable tabbing to the selected button.

Select a **Justification** mode to designate where the icon (if any) will appear in relation to the button text.

Control Customization

Throughout parts of the Clarion template wizards, there are default window controls that are automatically populated as part of a generated procedure. Each control has settings for the

X and Y, width and height parameters as follows:

X
Enter an integer constant that specifies the horizontal position of the top left corner of the control.

Y
Enter an integer constant that specifies the vertical position of the top left corner of the control.

Width
Enter an integer constant that specifies the width of the specified control

Height
Enter an integer constant that specifies the height of the specified control.

Justification
Select a **Justification** mode to designate how the text will appear in relation to the control's position.

Options

For Progress controls, you also have the option to designate a **vertical** (bottom to top) orientation, and a **smooth** bar display.

Form Customization

The settings on this window control the appearance of the Form procedure. These settings can be saved for use in other future applications.

Procedure Name

Enter a name that the Form Wizard will use to generate procedure names in your application. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for each file selected by the wizard. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names.

Window
Caption
Text that is entered on this line will be used as the description of the window used by the Form Wizard.

Background
Enter a default image here to use as a graphic or watermark for your Form window.

Mode
If you have designated a background image to use for your Form window, this option becomes available to control if the image is tiled, stretched, or centered.

Icon
Enter a default icon, or press the ellipsis button to select an icon file for use in your Form procedure's window. This will allow your Form window to be minimized if needed.

Font
Press the Font button to select a default font to use for the Form procedure. Sample text shown below the button is provided to allow you to review your selection.

Options
Select from the drop list to designate the initial **position** of your Form window. You can center the window, or use the default position that is set by the template wizard.

In addition, click on the **System Menu** check box to add the Windows System Menu to your Form procedure. If you will be using any entry fields on the Form window, you can also click on the **Entry Patterns** check box to allow special formatting picture information (Example: phone numbers or date pictures)

The **Entry Mode** drop list allows you to set the typing mode for the Form. Choose either **Insert**, **Overwrite**, or **Default**. The **Entry Mode** applies only for windows with the MASK attribute set. **Default** accepts input according to the current system settings.

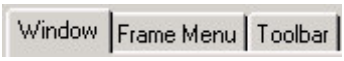
Tabs
Form Tabs
Enter text that will be used for each tab control that will be generated by the Form wizard. The default setting is ***&%#) %Text.*** The ampersand (&) identifies the next character as the hot key of the tab control. The **%#** macro identifies the instance number of the tab generated, determined by the number of fields per tab control and the total number of fields populated. The %Text is a declared template symbol that defaults to "General". You can remove this symbol and substitute it with any text that you wish.
Example: *2) General*

Buttons
The Buttons tab is the central control for all buttons used on the window. There is a default **Width** and **Height** setting (in dialog units) used for all buttons. However, each button can be overridden individually through the Button Customization dialog.

The default buttons of the Form wizard are:

OK
Cancel
Child
Help

Frame Customization



The settings on this window control the appearance of the Frame procedure. These settings can be saved for use in other future applications.

Window
Background
Enter a default image here to use as a graphic or watermark for your Frame procedure's client area (the area below your menu and toolbar by default).

Mode
If you have designated a background image to use for your Frame's client area, this option becomes available to control if the image is tiled, stretched, or centered.

Icon
Enter a default icon to use for your Frame procedure's window. This will allow your default applications to be minimized if needed.

Font
Press the Font button to select a default font to use for the Frame procedure. Sample text shown below the button is provided to allow you to review your selection.

Position
Select from the drop list to designate the initial position of your Frame window. You can center the window, or use the default position that is set by the template wizard.

In addition, click on the **System Menu** check box to add the Windows System Menu to your Frame procedure. If you will be using any entry fields on the Frame window's tool bar, you can also click on the **Entry Patterns** check box to allow special formatting picture information (Example: phone numbers or date pictures)

FrameMenu

File
Enable File Menu
Check this box if you would like the wizard to create a File menu in your Frame procedure. In addition, you can customize the following default menu items and their associated messages (which are displayed in the Frame's status bar by default):

File Menu Text
Print Item Text
Print Item Msg
Exit Item Text
Exit Item Msg

Edit
Enable Edit Menu
Check this box if you would like the wizard to create an Edit menu in your Frame procedure. In addition, you can customize the following default menu items and their associated messages (which are displayed in the Frame's status bar by default):

Edit Menu Text
Cut Item Text
Cut Item Msg
Copy Item Text
Copy Item Msg
Paste Item Text
Paste Item Msg

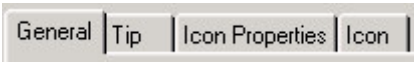
Window
Enable Window Menu
Check this box if you would like the wizard to create a standard Window menu in your Frame procedure. In addition, you can customize the following default menu items and their associated messages (which are displayed in the Frame's status bar by default):

Window Menu Text
Tile Item Text
Tile Item Msg
Cascade Item Text
Cascade Item Msg
Arrange Icons Item Text
Arrange Icons Item Msg

Help
Enable Help Menu
Check this box if you would like the wizard to create a standard Window menu in your Frame procedure. In addition, you can customize the following default menu items and their associated messages (which are displayed in the Frame's status bar by default):

Help Menu Text
Contents Item Text
Contents Item Msg
Search Item Text
Search Item Msg
How Item Text
How Item Msg

Toolbar



General
Background
Enter a default image here to use as a graphic or watermark for your Frame's toolbar.

Mode
If you have designated a background image to use for your Frame's toolbar, this option becomes available to control if the image is tiled, stretched, or centered.

Height
Enter the height in dialog units for the Frame procedure toolbar.

Tip
This tab identifies the toolbar buttons that are available from the wizard. You can modify the tool tip message for each button here.

Icon Properties
Width
The Frame procedure wizard has the option to produce toolbar navigation buttons. Enter here the default width of each button in dialog units.

Height
The Frame procedure wizard has the option to produce toolbar navigation buttons. Enter here the default height In dialog units of each button. Care must be taken not to exceed the default height of the overall toolbar properties.

Flat
Activate this checkbox to produce a flat button appearance for all toolbar buttons.

Icon
This tab identifies the toolbar buttons' default icon files used by the wizard. You can modify the icon files for each button here.

Process Customization

The settings here control the appearance of the Process template procedure generated by the Process Wizard. These settings can also be saved for use in other future applications.

Procedure Name
Enter a name that the Process Wizard will use to generate as a procedure name in your application. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for each file used by the wizard. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names. This name is used for files that have no keys defined

Proc. "With Key" Name
This prompt is similar to Procedure Name, but is used for designated files that have a key (or keys) defined in the Dictionary Editor.

Progress Window
Name
Text that is entered on this line will be used as the description of the window used by the Process Wizard.

Background
Enter a default image here to use as a graphic or watermark for your Process window.

Mode
If you have designated a background image to use for your Process window, this option becomes available to control if the image is *Tiled*, *Stretched*, or *Centered*.

Icon
Enter a default icon, or press the ellipsis button to select an icon file for use in your Process procedure's window. This will allow your Form window to be minimized if needed.

Font
Press the Font button to select a default font to use for the Process window and controls. Sample text shown below the button is provided to allow you to review your selection.

Options
Select from the drop list to designate the initial **Width, Height ,** and **Position** of your Process window. You can optionally center the window, or use the default position that is set by the template wizard.

Controls:

ProgressTPLWizardControlCustomization
Pct Text (Percentage control)TPLWizardControlCustomization
User String (user defined message)TPLWizardControlCustomization
Cancel ButtonButton_Customization

Process Control Customization

Throughout parts of the Clarion template wizards, there are default window controls that are automatically populated as part of a generated procedure. Each control has settings for the

X and Y, width and height parameters as follows:

X
Enter an integer constant that specifies the horizontal position of the top left corner of the control.

Y
Enter an integer constant that specifies the vertical position of the top left corner of the control.

Width
Enter an integer constant that specifies the width of the specified control

Height
Enter an integer constant that specifies the height of the specified control.

Justification
Select a **Justification** mode to designate how the text will appear in relation to the control's position.

Cancel ButtonTPLWizardCustomizationButton

Report Customization

The settings here control the appearance of the Report procedure generated by the Report Wizard. These settings can be saved for use in other future applications.

Procedure:

Procedure Name
Enter a name that the Report Wizard will use to generate procedure names in your application. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for each file selected by the wizard. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names. This name is used for files that have no keys defined

Procedure "With Key"
This prompt is similar to Procedure Name, but is used for designated files that have a key (or keys) defined in the Dictionary Editor.

Progress Window

Caption
Text that is entered on this line will be used as the description of the report's progress window, and appear in the progress window's title bar.

Background
Enter a default image here to use as a graphic or watermark for your report's progress window.

Mode
If you have designated a background image to use for your report's progress window, this option becomes available to control if the image is tiled, stretched, or centered.

Icon
Enter a default icon, or press the ellipsis button to select an icon file for use in your report's progress window. This will allow your window to be minimized if needed.

Font
Press the Font button to select a default font to use for the report's progress window. Sample text shown below the button is provided to allow you to review your selection.

Width
Enter the default width in dialog units for the report's progress window.

Height
Enter the default height in dialog units for the report's progress window.

Position
Select from the drop list to designate the initial position of your Progress window. You can center the window, or use the default position that is set by the template wizard.

Controls
There are four default controls in the Progress Window that you can customize:

ProgressTPLWizardControlCustomization

Pct TextTPLWizardControlCustomization

User StringTPLWizardControlCustomization

Cancel ButtonButton_Customization

Report Layout

General

The settings in this tab control refer to the settings of the current active report layout, whose name is displayed at the top.

Job Name
Enter the print job name to use for the Windows Print Manager. If omitted, the REPORT's *label* is used. The default setting is "Report %FileName".

Paper Type
Select the paper size for the report output from the drop list provided.

Paper Width
If you select "Other" as the Paper Type, you must enter a custom paper width.

Paper Height
If you select "Other" as the Paper Type, you must enter a custom paper height.

Margins
Margin setting control the printable area of your reports. Specify the Top, Bottom, Left and Right margin settings in thousandths of an inch

Check the Automatic Adjust Top and Bottom box to allow auto resizing of the margins when the report header or footer heights are resized.

Press the Adjust Using Header and Footer Size button to manually update the margin settings to conform to any changes made to the header and footer height positions. This button is only enabled if the Automatic Adjust Top and Bottom box is unchecked, to avoid sizing conflicts.

Orientation
Specify here what paper orientation the report layout will use (Portrait or Landscape).

Font
Press the Font Button to specify a default font to use for reports that the wizard will generate. There are several other places where you can override this setting.

Show Preview
Check this box to designate that all reports generated by the wizard will have a Print Preview window associated with it.

Preview Maximized
If the Show Preview box is checked, you can designate here that the Print Preview window will open in maximized mode (full screen).

Initial Zoom Setting
If the Show Preview box is checked, you can designate an initial zoom setting from this setting.

Header

The HEADER structure declares the output that prints at the beginning of each page or group.

Add Header
Check this box to allow your reports that are generated by the wizards to declare a HEADER section.

Background Color
If your user has support for color printing, you can designate a color to use as a background for the report's header.

Report Title
Enter a string or expression that will be used as the report title for each report generated by the report wizard. Since you can generate many report from different tables specified, it is a good idea to use a template macro here. The default is "Report *%FileName* file"

Position Y
Specify the starting position of the report title, relative to the start of the printable header area. Value is expressed in thousandths of an inch.

Justification
Specify how the contents of header text will be justified. The default is *Center*, but you can also specify *Left* or *Right* justified.

Title Font
The report title's font can be controlled with this setting. Press the *Title Font* button to designate the appearance of the title text.

Position
Enter the starting X and Y coordinates for the upper left corner of the header area, which is measured in thousandths of an inch, and relative to the upper left corner of the report's printable area. Enter the header area's width and height, also measured in thousandths of an inch

Font
Press the *Font* button to select the font (typeface), size, style (such as bold or italic), color, and font effects (underline and strikeout) for all controls in the header section. A sample of the selected font is displayed on the tab control.

Header Box
Check the Add Header box to add a box control to the report header. You will then be able to specify a color, Top (or width), and Height settings.

Detail

The detail area is the "body" of the report. Usually, your most important and relevant data will be printed here.

Background Color
If your user has support for color printing, press the ellipsis button to designate a color here to use as a background for the report's detail area.

Position
Enter the starting X and Y coordinates for the upper left corner of the detail area, which is measured in thousandths of an inch, and relative to the upper left corner of the report's printable area, or to the last item printed in the detail print area. Enter the detail area's width and height, also measured in thousandths of an inch.

Think of this setting as the "record" or "row" area of your report.

Font
Press the *Font* button to select the font (typeface), size, style (such as bold or italic), color, and font effects (underline and strikeout) for all controls in the Detail section. A sample of the selected font is displayed on the tab control.

Add Detail Lines
Check this box to instruct the template to generate a horizontal line between each detail section printed.

Automatic adjust Band Height with Row Number
Check this box to instruct the template to generate the total detail area's height based on the number of rows that are calculated at design time. The number of rows calculated is based on the data elements selected to print and number of report columns designated for each row.

This prevents extra white space from being generated between the last detail on the page and the page footer.

Footer

The report FOOTER area is normally used to print text and data at the end of each page.

Add Footer
Check this box to allow your reports that are generated by the wizards to declare a FOOTER section.

Background Color
If your user has support for color printing, press the ellipsis button to designate a color here to use as a background for the report's footer area.

Position
Enter the starting X and Y coordinates for the upper left corner of the footer area, which is measured in thousandths of an inch, and relative to the *lower left corner of the report's detail area*, or to the last item printed in the detail print area. Enter the detail area's width and height, also measured in thousandths of an inch.

Font
Press the *Font* button to select the font (typeface), size, style (such as bold or italic), color, and font effects (underline and strikeout) for all controls in the report's footer section. A sample of the selected font is displayed on the tab control.

Form
The Form Band is normally used to specify constant text or graphics which print on every page (for example, a watermark or a tax form)

Add Form
Check this box to allow your reports generated by the wizards to specify a Form section.

Background Color
If your user has support for color printing, press the ellipsis button to designate a color here to use as a background for the report's form band area.

Margins
Margin settings control the printable area of your form band. Specify the Top, Bottom, Left and Right margin settings in thousandths of an inch

Font
Press the *Font* button to select the font (typeface), size, style (such as bold or italic), color, and font effects (underline and strikeout) for all controls in the report's Form Band section. A sample of the selected font is displayed on the tab control.

Image

Add Image
Check this box to allow an image control to be populated on the Form band. This check box also enables the remaining options.

Image File
Press the ellipsis button to select the name of the image file to be used in the populated image control. Any image format supported by the Windows print engine may be selected.

Mode
Select from the drop list how the image file will be rendered on the report. Select from tiled, stretched, and centered.

Image Position
If width and height are zero(0), the image will use the settings specified by the form band. Set the X and Y position to designate the anchor position of the top left corner.

Additions

Page Number ControlReport_Additions_Customization

Date and Time ControlReport_Additions_Customization

Report Label Customization

The settings here control the appearance of the Report procedure with a Label layout generated by the Label Wizard. These settings can be saved for use in other future applications.

Procedure

Procedure Name
Enter a name that the Label Wizard will use to generate procedure names in your application. The template macro symbol, %FileName, is required, and extracts the Dictionary file name for each file selected by the wizard. You can modify this line with other template macros and text if you wish. No spaces are allowed in procedure names. This name is used for files that have no keys defined

Proc. "With Key" Name
This prompt is similar to Procedure Name, but is used for designated files that have a key (or keys) defined in the Dictionary Editor.

Progress Window

Name
Text that is entered on this line will be used as the description of the report's progress window, and appear in the progress window's title bar.

Background
Enter a default image here to use as a graphic or watermark for your report's progress window.

Mode
If you have designated a background image to use for your report's progress window, this option becomes available to control if the image is tiled, stretched, or centered.

Icon
Enter a default icon, or press the ellipsis button to select an icon file for use in your report's progress window. This will allow your window to be minimized if needed.

Font
Press the Font button to select a default font to use for the report's progress window. Sample text shown below the button is provided to allow you to review your selection.

Width
Enter the default width in dialog units for the report's progress window.

Height
Enter the default height in dialog units for the report's progress window.

Position
Select from the drop list to designate the initial position of your Progress window. You can center the window, or use the default position that is set by the template wizard.

Controls
There are four default controls in the Progress Window that you can customize:

Progress

Pct Text

User String

Cancel Button

Report Layout

General
The settings in this tab control refer to the settings of the current active report layout, whose name is displayed at the top.

Job Name
Enter the print job name to use for the Windows Print Manager. If omitted, the REPORT's *label* is used. The default setting is "%FileName Report".

Orientation
Specify here what paper orientation the report layout will use (Portrait or Landscape).

Font
Press the Font Button to specify a default font to use for reports that the wizard will generate. There are several other places where you can override this setting.

Show Preview
Check this box to designate that all reports generated by the wizard will have a Print Preview window associated with it.

Preview Maximized
If the Show Preview box is checked, you can designate here that the Print Preview window will open in maximized mode (full screen).

Initial Zoom Setting
If the Show Preview box is checked, you can designate an initial zoom setting from this setting.

Background Color
If your printer has support for color printing, you can designate a color to use as a background for the printed labels.

Font
Press the *Font* button to select the font (typeface), size, style (such as bold or italic), color, and font effects (underline and strikeout) for all controls in the label section. A sample of the selected font is displayed on the tab control**.**

Label Size

In most cases, the report label settings, based on the Label Group and Label Type, will be preset for you. Use this tab control's settings if you have specified a **Custom** Label Group or **Custom** Label Type. The graphic provided on this tab control guides you through the parameters available here.

All measurements are in thousandths of an inch by default.

Width
Enter the width of the individual label's printable area.

Height
Enter the width of the individual label's printable area.

Top margin
Enter the height of the label's non-printable area, measured from the top of the page. For example, if you want the first label to print one inch from the top of the page, enter 1000.

Left margin
Enter the height of the label's non-printable area, measured from the left edge of the page. For example, if you want the first label to print one-half inch from the left edge, enter 500.

Number
Enter the number of labels to print across the page.

Number down
Enter the number of labels to print down the page.

Horizontal
Enter the horizontal distance between labels, measured from the left edge of one label to the left edge of the next label.

Vertical Pitch
Enter the vertical distance between labels, measured from the left edge of one label to the left edge of the next label.

Paper Type
Select a paper size from the drop list corresponding to the page size that you will print labels on. If you select **Other**, you can customize the page size to any length and height, using the **Paper Width** and **Paper Height** settings.

Report Additions Customization

Date and Time

Add Date and Time
Check this box to include an automatic Report Date and Time template control on your reports generated by the wizards.

Position
Specify which report band to populate the Date and Time controls. Select from *Header*, *Footer*, *Detail*, *Form*, and **both** *Header and Footer*

Date and Time Pictures
Press the ellipsis button on the appropriate Date or Time entry to call up the *Edit Picture* dialog window, and build a picture token to use for each one. The default picture is *@D17* and *@T7* respectively.

Date and Time Positions
Specify the exact position within the selected bands where the date and time controls and prompts will be populated.

Page Number

Add Page Number
Check this box to include an automatic Page Number template control on your report.

Position
Specify which report band to populate the Page Number control. Select from *Header*, *Footer*, *Detail*, *Form*, and **both** *Header and Footer*

Page Picture
Press the ellipsis button to call up the *Edit Picture* dialog window, and build a picture token to use for the page number. The default picture is *@pPage <<#p*

Pos X and Pos Y
Specify the exact position within the selected bands where the control will be populated.

Save As Theme

Create a new theme?
Check this box to write your current changes to a new theme name and associated external file.

Save the choices that you made?
Check this box to preserve your settings on this window. For example, if you always want to save the report layout with the theme, check this box.

Theme Name
Enter a name that will be used in the Theme Selection Drop List for all future wizard use.

File Name
Enter the physical file name that will be used to write your current changes to. Although it is not required, it is probably a good idea to keep the file name consistent with the Theme name entered above.

Change Layout Name?
Check this box to write the current report layout changes to a new layout name.

New Layout Name
Enter the new layout name here.

Save It
Press this button to write the current changes to the theme file name entered above.

Press the OK button to maintain your current checkbox settings on this window, or press the Cancel button to reset them to their original values when you entered this window.

Save Theme

Save the choices that you made?

Check this box to preserve your settings on this window. For example, if you always want to save the report layout with the theme, check this box.

Change Layout Name?

Check this box to allow a new layout name to be saved. If you leave this box unchecked, the current report layout name will be saved with any changes you have made.

Save It
Press this button to write the current changes to the *Default.TFT* theme file.

Press the **OK** button to maintain your current checkbox settings on this window, or press the **Cancel** button to reset them to their original values.

Sort Order Customization

This window is used to set the sort order style of a default Browse procedure. Sort order style is defined as the way you navigate from one sort value to the next based on the number of keys that you designate for the Browse. You can select from the following options:

Which selection style?
Choose from *Tabs*, *Drop List*, or *Popup menu*.

The *Tabs* selection uses tab controls to signal the list box that a new sort order is to be applied. The *Drop List* option populates a drop list control that contains the sort order selections of the selected Browse Box control. The *Popup menu* option stores the sort order selections in a popup menu, which is accessed when right-clicking on the Browse Box (List Box) control, or pressing the Sort Order button that is created. In addition, *Popup menu* enables the following additional options for the Sort Order button:

Text
Enter the default text to display on the Sort Order button.

Icon
Enter an icon file or equate that you wish to use for the Sort Order button.

Cursor
Enter a default cursor file or equate that you wish to use for the Sort Order button. This cursor will be displayed when the mouse moves over the button area.

HotKey
Enter a default hot key to use to automatically call the Sort Order button.

Message
Enter a default message to use for the Sort Order button. This text will appear by default in the window's status bar.

Tip
Enter a tool tip message for the Sort Order button. This text will appear when the mouse is moved over the button.

Options
Activate the **Flat** check box to give the Sort Order button a flat appearance.

Activate the **Skip** check box to disable tabbing to the Sort Order button.

Select a **Justification** mode to designate where the icon (if any) will appear in relation to the button text.

Theme Design

Wizard Prompts

The Prompts buttons allow default settings to be established for the selected theme. Each prompt is described below.

Application

Control ModelWizard__Control_model

Reports
Check the Generate Reports for each file box to automatically generate report procedures for this selected theme.

Browse

Call Update Procedure
Check this box to allow this theme to always generate a Form (Update) procedure for each Browse that is generated.

Child Files
If the browse wizard is accessing a parent table, check this box to allow a button to be generated for each child file defined in the Data Dictionary.

Parent File Handling
If the browse wizard is accessing a child table, you can specify how the browse will process the records:

Do not select by parent record
Do not limit the browse – browse all records.

Select parent record via button
Browse only the child records for a specific parent record. Provide a button to select the parent record.

Assume that the parent record is active
Browse only the child records for a specific parent record. Assume the parent record is already active, and do not provide a button.

Select Button
Check this box to provide a "Select" button that displays when the Browse procedure is called to select a record, but is hidden when the Browse is called to update records.

Form

Updating Records
Enable the appropriate check boxes to allow the wizard to include a Save Button extension to any Form that uses the selected theme.

MessagesTPLControlSaveButton_Messages_and_Titles

Record Validation
Enable the appropriate check boxes to allow the wizard to include the Record Validation extension to any Form that uses the selected theme.

Child File Handling
If the primary table to be updated by the Form is defined as the parent in a parent/child relationship defined in the Data Dictionary, you can specify this theme to handle the child file in one of the following ways:

Place children on tabs
Access children on push buttons
Do not provide child access

Report

Sort Order

Select from the drop list the sort and report generation method from the following choices:

Single Key

Select this option to force the wizard to generate a separate report for the key that you select in the Enter a key prompt that follows.

Runtime Key Selection

Select this option to force the wizard to generate a single report that pops up a sort order dialog prior to printing at runtime.

Record Order

Select this option to force the wizard to generate a single report sorted by record order for your selected file (or files).

Columns
How many columns do you want the report to use? Type the number of columns for your report. The Report Wizard distributes the report columns evenly across the columns.

Customization:

Frame Customization

Form Customization

Process Customization

Browse Customization

Report Customization

Window Customization

Theme Maintenance Wizard

The Theme Maintenance Wizard is a powerful template utility that allows you to Add, Modify, or Delete selected template wizard themes.

Themes are defined as text files that control a wizard's generated output and, in addition, the wizard's default prompt settings that are presented to the developer. A theme file has a default file extension of TFT, and is stored in the TFT sub folder of the Clarion TEMPLATE folder. This path, and the list of existing themes, is found in the C7TFT.INI file located in the Clarion BIN folder.

The Theme Maintenance Wizard contains the following options:

Theme Selection

Theme	From the drop down list, select an existing theme that you wish to add, modify, or delete. If you are adding a new theme, you will need to select an existing theme. After you make modification to the selected theme, you can use the Save As option.
Report Layout	From the drop down list, select an existing report layout that you wish to add, modify, or delete. Changes to the Report Wizard will be applied to this layout.

Operations

The buttons shown on this wizard sheet direct you to one of four operations. You can **Design Theme**, based on the theme selected on the previous window. After your modifications, return to this window to **Save Theme Default.TFT** (the selected theme), or **Save As** to a new theme name.

After saving the theme through one of the above methods, press the **Cancel** button to exit the wizard. Your theme is already saved. You can also press the **Finish** button to exit the wizard, but you will have to complete the required entries in the **Save As** dialog.

You can also remove the selected theme permanently by pressing the **Delete Theme Default.TFT** button.

Window Customization

Window

Name
Enter a name to use as the default caption, or title, for all windows generated.

Background
Enter a default image here to use as a graphic or watermark for your Window procedure control area.

Mode
If you have designated a background image to use for your Window procedure, this option becomes available to control if the displayed image is tiled, stretched, or centered.

Icon
Enter a default icon to use for your Window procedure. This will allow your window to be minimized if needed.

Font
Press the Font button to select a default font to use for the Window procedure. Sample text shown below the button is provided to allow you to review your selection.

Position
Select from the drop list to designate the initial position of your window. You can optionally center the window, or use the default position that is set by the template wizard.

In addition, click on the **System Menu** check box to add the Windows System Menu to your Window procedure. If you will be using any special entry control pictures on the window, you can also click on the **Entry Patterns** check box to allow the display of special formatting picture information (Example: phone numbers or date pictures)

Buttons

The Window Wizard has three optional buttons that you can choose to populate. They are the **OK**, **Cancel** and **Help** button controls. Check on each "Enable *name* button" box to activate the appropriate button dialogs. More information on each button's prompts are located here.

Additional Libraries and Templates

This section references additional libraries and templates that are not a part of the ABC Library or standard shipping templates:

Crystal Report Templates

Crystal8 Class

Finance Library

HTML Help Templates - Overview

MenuStyleManager Class

Rich Text Template Support - Overview

Statistics Library

Crystal Reports Templates

Crystal Reports Overview

Business Objects's Crystal Reports is one of the leading report writers delivering Windows reports. For more information on this product visit the Crystal Reports web site at www.businessobjects.com.

Clarion's Crystal Report interface is comprised of templates, libraries, and DLLs that communicate with Crystal Reports, versions 8 or 9 (see below). The DLL is accessed by a Class Interface and is hooked to your application using simple standard Clarion code. This interface allows a seamless integration of previously defined Crystal reports within a Clarion application. The Crystal report engine accesses data and creates the report. The report can be previewed in a Clarion window.

Clarion's Crystal Reports implementation is compatible with both the ABC and Legacy templates. It can only be used in 32-bit applications.

Crystal Reports Files

The files that make up Clarion's Crystal Reports interface are:

C70cr8.inc Crystal Class Definition
C70cr8L.inc Crystal Class Definition Local Compile
C70cr8.tpl Templates
C70cr8.lib Crystal LIB
C70cr8L.lib Crystal Local LIB
C70cr8.dll Crystal DLL

Files to Distribute

To distribute an application that includes the Crystal Report interface, the following files must be distributed.

C70cr8.dll Crystal DLL

This DLL file must be included in distribution when an application is compiled using the Standalone (C70RUNx.DLL) Run-Time library.

Other Runtime Distribution Files

According to the latest Crystal Reports documentation (Version 9), the required runtime files for an application developed using the Crystal Report Print Engine API (crpe32.dll) are listed below.

The Crystal Reports Print Engine is now considered a legacy API and will no longer expose calls for any the new features included in Crystal Reports. For more information, see "Retired Developer APIs" in *Crystal Reports Developer's Guide (CrystalDevHelp.chm)*.

File Developer/Distribution Locations Description
(Locations are the same unless listed separately.)

crpe32.dll \Program Files\Common Files\Crystal Decisions\2.0\bin Interface to the Crystal Report Engine

crqe.dll \Program Files\Common Files\Crystal Decisions\2.0\bin Crystal Query Engine

Implode.dll \Program Files\Common Files\Crystal Decisions\2.0\bin Compression Library

msvcrt.dll \Windows\system32 or \WINNT\system32 (NT only) Microsoft (R) C Runtime Library

querybuilder.dll \Program Files\Common Files\Crystal Decisions\2.0\bin Crystal Query Builder

riched20.dll \Windows\system32 or \WINNT\system32 (NT only) Rich Text Edit Control, v3.0

ufmanager.dll \Program Files\Common Files\Crystal Decisions\2.0\bin Manager for loading UFL's

unicows.dll See the note below. Unicode Layer for Win9x

usp10.dll \Windows\system32 or \WINNT\system32 (NT only) Uniscribe Unicode script processor

Note:

The following DLLs need only be included under these circumstances:

Include Implode.dll only for applications using reports created in versions previous to Crystal Reports 8.

Include unicows.dll only for applications installed on Win9x machines. Due to licensing restrictions, you must get unicows.dll from the Microsoft web site.

Go to http://www.microsoft.com/msdownload/platformsdk/sdkupdate/default.htm? and select "Microsoft Layer for Unicode on Windows 95/98/ME Systems (MSLU) version 1.0".

If your application uses any of the following functions,

CrPEExportToDisk CrPEExportToExch crPEExportToMapi

CrPEExportToHTML CrPEExportToODBC crPESelectPrinter

CrPEGetSelectedPrinter

you must include:

crwrap32.dll \Program Files\Common Files\Crystal Decisions\2.0\bin
Print Engine Wrapper (intended as a sample of how to prototype structured calls for languages that do not support structures)

See your Crystal Reports Documentation for any additional runtime requirements.

Crystal Reports Code Templates

Code templates generate source code into an embed point that you specify and sometimes direct source generation into other embed points as well. Their purpose is to make procedure customization quick and easy. Each Crystal Report Code template provides a well-defined task. For example, the PreviewCrystalReport template simply provides a way for previewing and printing of predefined Crystal Reports. Typically, a Code template provides a dialog box with prompts and instructions.

PreviewCrystalReport

PrintCrystalReport

GetCrystalFormulaPreview

GetCrystalFormulaPrint

GetCrystalQueryPreview

GetCrystalQueryPrint

SetCrystalFormulaPreview

SetCrystalFormulaPrint

SetCrystalQueryPreview

SetCrystalQueryPrint

GetCrystalFormulaPreview / GetCrystalFormulaPrint

The **GetCrystalFormulaPreview** and **GetCrystalFormulaPrint** templates both accomplish the exact same task. They retrieve the Formula that is used to limit retrieved records. The difference between these two templates is that **GetCrystalFormulaPreview** is used in conjunction with the **PreviewCrystalReport** template and **GetCrystalFormulaPrint** is used in conjunction with the **PrintCrystalReport** template.

These templates call the Crystal8.SelectionFormula method.

Requirements

These templates require the **PreviewCrystalReport** or the **PrintCrystalReport** template respectively.

Populating the Template

1. From an embed point, press the **Insert** button and select either the **GetCrystalFormulaPreview** or **GetCrystalFormulaPrint** template.

Template Prompts

These templates provides the following prompts:

Target Variable
Specify a valid variable to store the retrieved Formula. This may be a STRING, CSTRING, PSTRING, or MEMO data type. The variable may be a local, module, or global variable. You may also use a file field; however, you must write the code to update the file. This variable is required.

GetCrystalQueryPreview / GetCrystalQueryPrint

The **GetCrystalQueryPreview** and **GetCrystalQueryPrint** templates both accomplish the exact same task. They retrieve the SQL Query that is sent to the SQL data source. The difference between these two templates is that **GetCrystalQueryPreview** is used in conjunction with the **PreviewCrystalReport** template and **GetCrystalQueryPrint** is used in conjunction with the **PrintCrystalReport** template.

These templates call the Crystal8.Query method. For more information on these methods refer to the *ABC Library Reference, Crystal8 class.*

Requirements

There templates require the **PreviewCrystalReport** or the **PrintCrystalReport** template respectively.

Populating the Template

1. From an embed point, p**ress** the **Insert** button and select either the **GetCrystalQueryPreview** or **GetCrystalQueryPrint** template.

Template Prompts

These templates provides the following prompts:

Target Variable
Specify a valid variable to store the retrieved SQL query. This may be a STRING, CSTRING, PSTRING, or MEMO data type. The variable may be a local, module, or global variable. You may also use a file field; however, you must write the code to update the file. This variable is required.

PreviewCrystalReport

This template allows for previewing and printing of predefined Crystal Reports.

Requirements

There are no requirements for this template.

Populating the Template

1. From an embed point, press the **Insert** button and select the **PreviewCrystalReport** template.

Template Prompts

The PreviewCrystalReport template provides the following prompts:

General

Object Name
Set the object's label for the template-generated code. By fine-tuning the object names, you can make your generated code easier to read. The default object name is oCrystal8.

Report Name or variable (prefixed with !)
Type a valid Crystal report name including extension (.rpt) or a variable that will contain the report name. When using a variable, precede it with an exclamation (!). The report name must include the runtime path of the file. If the file is not found at runtime an error window will display containing the following error- "error opening job".

Window Title or variable (prefixed with !)
Crystal Reports are run inside a Clarion window. Type a title to appear in the window caption bar or enter a variable name which will contain the window title. When using a variable, precede it with an exclamation (!).

Show Print Controls?
Check this box to turn off all preview window buttons. If checked, the **Control Options** template tab will not be available.

Control Options

This tab will not show if the **Show Print Controls?** option on the **General** tab is not checked.

Allow Prompting?
Check this box to allow the Crystal Report use its defined runtime parameter fields.

Allow Drill Down?
Check this box to allow support of Crystal's drill-down report feature.

Show Cancel Button?
Check this box to enable a **Cancel** button on the report preview window.

Show Close Button?
Check this box to enable a **Close** button on the report preview window.

Show Export Button?
Check this box to enable an **Export** button on the report preview window.

Show Launch Button?
Check this box to enable a **Launch** button on the report preview window. The Launch button runs the Analysis tool.

Show Navigation Buttons?
Check this box to enable page navigation buttons on the report preview window.

Show Print Button?
Check this box to enable a **Print** button on the report preview window.

Show Print Setup?
Check this box to enable a **PrintSetup** button on the report preview window.

Show Progress?
Check this box to enable the report progress controls in the preview window. These controls include Total Records, Records Read, Percent Read.

Show Refresh?
Check this box to enable a **Refresh** button on the report preview window.

Show Search?
Check this box to enable the two search controls on the report preview window. These two controls include a search button and an entry control.

Show Zoom?
Check this box to enable the zoom control on the report preview window.

Show Toolbar Tips?
Check this box to enable tool tips on the toolbar in the preview window.

Show Document Tips?
Check this box to enable tool tips on the document (report) in the preview window.

Window Options

Initial State
Select the initial size and state of your window from the drop-down list. Choose from:

Normal Display the window at the default size. If you don't specify a default size, Clarion's run-time library sets it for you.

Maximized This window fills the entire desktop, or the entire application frame, depending on whether the window is an application window, or an MDI child window.

Iconized The window appears in an iconized state in the Taskbar.

Frame Type
Pick the frame type for your window from the drop-down list. Choose from:

Single A single pixel frame that the user *cannot* resize. Most suitable for dialog boxes.

Double A thick frame, which the user *cannot* resize. Use this type frame for a system modal window (without a caption bar), or for a modal dialog box (with a caption bar).

Resizable A thick frame, which the user *may* resize. Choose this for application and MDI child windows.

None A single pixel frame. Most suitable for dialog boxes. The user cannot resize this frame.

Icon
To associate an icon with the window (and add a system menu—see **System Menu** below), specify an icon file name (.ICO file) in this field. Type the file name or press the ellipsis button (...) to select a file name with the standard **Open File** dialog. Specifying an icon automatically places a minimize button on the caption bar of your application or MDI child window.

System Menu
To place a system menu in your window, check the **System Menu** box, or specify an icon file (see **Icon** above), or specify a maximize box (see **Maximize** below). When your window has the SYSTEM attribute Windows 95/98 and Windows NT display an icon in the upper left corner. If you specify an icon (see **Icon** above), that icon displays, otherwise the system default icon displays. Initially, the system default icon is set to the Clarion icon; however, you can specify a system default icon with:

```
System{PROP:Icon} = 'My.ico'
```

Activate the system menu by CLICKING the button, box, or icon in the upper left corner of the window. Standard system menu choices include Restore, Minimize, Maximize, and Close.

Every application frame should have a system menu. For users on a system without a mouse, the system menu provides the only means of minimizing, maximizing or re-sizing the application window.

Maximize Box

To place a maximize button in your window (and a system menu—see **System Menu** above), check this box. In general, you should place a maximize button on application windows and MDI child document windows, not on dialog boxes.

3D Look

To provide the gray window background, and chiseled control look for your window, check this box. This is clearly a style consideration, but will go a long way in giving your application a professional look.

PrintCrystalReport

This template allows for printing of predefined Crystal Reports. The report cannot be previewed with this template. An optional printer setup dialog is available.

Requirements

There are no requirements for this template.

Populating the Template

1. From an embed point, press the **Insert** button and select the **PrintCrystalReport** template.

Template Prompts

The PrintCrystalReport template provides the following prompts:

Object Name
Set the object's label for the template-generated code. By fine-tuning the object names, you can make your generated code easier to read. The default object name is oCrystal8.

Report Name or variable (prefixed with !)
Type a valid Crystal report name including extension (.rpt) or a variable that will contain the report name. When using a variable, precede it with an exclamation (!). The report name must include the runtime path of the file. If the file is not found at runtime an error window will display containing the following error- "error opening job".

Number of Copies or variable
Specify the number of copies to print as a hard coded numeric value or specify a variable (prefixed with !) to set the number of copies at runtime.

Show Printer Setup?
Check this box to show the printer setup dialog before the report is sent to the printer. This allows the user to select a specific printer and set printer properties.

SetCrystalFormulaPreview / SetCrystalFormulaPrint

The **SetCrystalFormulaPreview** and **SetCrystalFormulaPrint** templates both accomplish the exact same task. They set the Formula that is used to limit retrieved records. The difference between these two templates is that **SetCrystalFormulaPreview** is used in conjunction with the **PreviewCrystalReport** template and **SetCrystalFormulaPrint** is used in conjunction with the **PrintCrystalReport** template.

These templates call the Crystal8.SelectionFormula method. For more information on these methods refer to the *ABC Library Reference, Crystal8 class.*

Requirements

There templates require the **PreviewCrystalReport** or the **PrintCrystalReport** template respectively.

Populating the Template

1. From an embed point, press the **Insert** button and select either the **SetCrystalFormulaPreview** or **SetCrystalFormulaPrint** template.

Template Prompts

These templates provides the following prompts:

Formula Variable

Specify a valid variable that holds the Formula to send to the report. This variable may be a STRING, CSTRING, PSTRING, or MEMO data type. The variable may be a local, module, or global variable. You may also use a file field. This variable is required.

SetCrystalQueryPreview / SetCrystalQueryPrint

The **SetCrystalQueryPreview** and **SetCrystalQueryPrint** templates both accomplish the exact same task. They set the SQL Query that wiil be sent to the SQL data source. The difference between these two templates is that **SetCrystalQueryPreview** is used in conjunction with the **PreviewCrystalReport** template and **SetCrystalQueryPrint** is used in conjunction with the **PrintCrystalReport** template.

These templates call the Crystal8.Query method. For more information on these methods refer to the *ABC Library Reference, Crystal8 class.*

Requirements

These templates require the **PreviewCrystalReport** or the **PrintCrystalReport** template respectively.

Populating the Template

1. From an embed point, p**ress** the **Insert** button and select either the **SetCrystalQueryPreview** or **SetCrystalQueryPrint** template.

Template Prompts

These templates provides the following prompts:

Query Variable

Specify a valid variable that holds the SQL Query to send to the SQL data source. This may be a STRING, CSTRING, PSTRING, or MEMO data type. The variable may be a local, module, or global variable. You may also use a file field. This variable is required.

Crystal8 Class

Business Object's Crystal Reports is one of the leading report writers delivering Windows reports. For more information on this product, visit the Business Objects web site at www.businessobjects.com.

Clarion's Crystal Report interface is comprised of templates, libraries, and DLLs that communicate with Crystal Reports, version 8 or 9. The DLL is accessed by a Class Interface and is hooked to your application using simple standard Clarion code. This interface allows a seamless integration of previously defined Crystal reports within a Clarion application. The Crystal report engine accesses data and creates the report. The report can be previewed in a Clarion window.

Clarion's Crystal Reports implementation is compatible with both the ABC and Legacy templates. It can only be used in 32-bit applications.

Crystal8 Class Concepts

Clarion's Crystal Reports implementation is a DLL that communicates with Business Object's Crystal Reports report writer. The Crystal 8 Class accesses the DLL. There are several templates available which make the interface to the report writer easily accessible from your Clarion program. Previewing and/or printing reports are simple.

Relationship to Other Application Builder Classes

The Crystal8 class works independently of all other ABC classes.

ABC Template Implementation

The PreviewCrystalReport and PrintCrystalReport template extensions instantiate an object based on the object name specified by either of these extensions. The object is instantiated in the procedure where the extension exists.

Crystal8 Source Files

The Crystal8 class declarations are installed by default to the Clarion \LIBSRC folder. The Crystal8 component is distributed as a LIB/DLL, therefore the source code for the methods is not available. However, the methods are defined in this chapter and may be implemented in applications provided the required LIB/DLL is available at runtime.

C70cr8.INC Crystal Class Definition
C70cr8l.INC Crystal Class Definition Local Compile
C70cr8.dll Crystal DLL
C70cr8.lib Crystal LIB
C70cr8L.lib Crystal Local LIB

Crystal8 Class Properties

There are no properties associated with the Crystal8 Class Library.

Crystal8 Methods

AllowPrompt (prompt for runtime parameter data)
CanDrillDown(allow Crystal drill down support)
HasCancelButton (display cancel button on report preview)
HasCloseButton (display close button on report preview)
HasExportButton (display export button on report preview)
HasLaunchButton (display launch button on report preview)
HasNavigationControls (display navigation controls on report preview)
HasPrintButton (display print button on report preview)
HasPrintSetupButton (display print setup button on report preview)
HasProgressControls (display progress controls on report preview)
HasRefreshButton (display refresh button on report preview)
HasSearchButton (display search button on report preview)
HasZoomControl (display zoom control on report preview)
Init (initialize Crystal8 object)
Kill (shut down Crystal8 object)
Preview (preview a Crystal Report)
_Print (print a Crystal Report)
Query (retrieve or set the SQL data query)
SelectionFormula (retrieve or set the Crystal formula)
ShowToolbarTips (show tips on preview window toolbar)
ShowDocumentTips (show tips on docuement in the preview window)
ShowReportControls (show print controls)

AllowPrompt (prompt for runtime parameter data)

AllowPrompt(| *allowpromptflag* |)

AllowPrompt Allow report runtime prompting for Crystal Parameter fields.

allowpromptflag An integer constant, variable, EQUATE, or expression that specifies whether the report will prompt for runtime parameter fields. A value of one (1) is used to allow prompting; a value of zero (0) is used to disallow field prompting.

The AllowPrompt method can conditionally allow the Crystal report that is being previewed or printed to prompt for runtime parameter fields. These parameter fields must be defined in the Crystal report. This method returns a BYTE representing the value of *allowpromptflag*.

Return Data Type: BYTE

Example:

```
oCrystal8.AllowPrompt(1)
```

CanDrillDown(allow Crystal drill down support)

CanDrillDown(| *candrilldown* |)

CanDrillDown Allows use of Crystal Report's drill down feature.

candrilldown An integer constant, variable, EQUATE, or expression that specifies whether the report make use of Crystal's drill down feature. A value of one (1) allows drill down to be used; a value of zero (0) removes the ability to drill down.

The CanDrillDown method allows a Crystal Report to use the defined drill down support. For more information on Crystal's drill down feature refer to the Crystal Report documentation. This method returns a BYTE representing the value of *candrilldown*.

Return Data Type: BYTE

Example:

```
oCrystal8.CanDrillDown(1)
```

HasCancelButton (display cancel button on report preview)

HasCancelButton (| *hascancelbutton* |)

HasCancelButton Allow a cancel button on the report preview window.

hascancelbutton An integer constant, variable, EQUATE, or expression that specifies whether a cancel button will appear on the report's preview window. A value of one (1) displays the cancel button; a value of zero (0) does not display the cancel button.

The HasCancelButton method is used to optionally display a cancel button on the report preview window. This method returns a BYTE representing the value of *hascancelbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasCancelButton(1)
```

HasCloseButton (display close button on report preview)

HasCloseButton (| *hasclosebutton* |)

HasCloseButton Allow a close button on the report preview window.

hasclosebutton An integer constant, variable, EQUATE, or expression that specifies whether a close button will appear on the reports preview window. A value of one (1) displays the close button; a value of zero (0) does not display the close button.

The HasCloseButton method is used to optionally display a close button on the report preview window. This method returns a BYTE representing the value of *hasclosebutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasCloseButton(1)
```

HasExportButton (display export button on report preview)

HasExportButton (| *hasexportbutton* |)

HasExportButton Allow an export button on the report preview window.

hasexportbutton An integer constant, variable, EQUATE, or expression that specifies whether an export button will appear on the reports preview window. A value of one (1) displays the export button; a value of zero (0) does not display the export button.

The HasExportButton method is used to optionally display an export button on the report preview window. This method returns a BYTE representing the value of *hasexportbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasExportButton(1)
```

HasLaunchButton (display launch button on report preview)

HasLaunchButton (| *haslaunchbutton* |)

HasLaunchButton Allow a launch button on the report preview window.

haslaunchbutton An integer constant, variable, EQUATE, or expression that specifies whether a launch button will appear on the reports preview window. A value of one (1) displays the launch button; a value of zero (0) does not display the launch button.

The HasLaunchButton method is used to optionally display a launch button on the report preview window. This method returns a BYTE representing the value of haslaunchbutton. The launch button is used to launch the Analysis tool.

Return Data Type: BYTE

Example:

```
oCrystal8.HasLaunchButton(1)
```

HasNavigationControls (display navigation controls on report preview)

HasNavigationControls (| *hasnavigationcontrols* |)

HasNavigationControls Allows navigation controls on the report preview window.

hasnavigationcontrols An integer constant, variable, EQUATE, or expression that specifies whether the navigation controls will appear on the report's preview window. A value of one (1) displays the navigation controls; a value of zero (0) does not display the navigation controls.

The HasNavigationControls method is used to optionally display navigation controls on the report preview window. This method returns a BYTE representing the value of *hasnavigationcontrols*. Navigation controls are used to navigate through a report, immediately to the beginning, end or anywhere in between.

Return Data Type: BYTE

Example:

```
oCrystal8.HasNavigationControls(1)
```

HasPrintButton (display print button on report preview)

HasPrintButton (| *hasprintbutton* |)

HasPrintButton Allows a print button on the report preview window.

Hasprintbutton An integer constant, variable, EQUATE, or expression that specifies whether a print button will appear on the report's preview window. A value of one (1) displays the print button; a value of zero (0) does not display the print button.

The HasPrintButton method is used to optionally display a print button on the report preview window. This method returns a BYTE representing the value of *hasprintbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasPrintButton(1)
```

HasPrintSetupButton (display print setup button on report preview)

HasPrintSetupButton (| *hasprintsetupbutton* |)

HasPrintSetupButton Allows a print setup button on the report preview window.

hasprintsetupbutton An integer constant, variable, EQUATE, or expression that specifies whether a print setup button will appear on the reports preview window. A value of one (1) displays the print setup button; a value of zero (0) does not display the print setup button.

The HasPrintSetupButton method is used to optionally display a print setup button on the report preview window. This method returns a BYTE representing the value of *hasprintsetupbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasPrintSetupButton(1)
```

HasProgressControls (display progress controls on report preview)

HasProgressControls (| *hasprogresscontrols* |)

HasProgressControls Allows navigation controls on the report preview window.

Hasprogresscontrols An integer constant, variable, EQUATE, or expression that specifies whether the progress controls will appear on the reports preview window. A value of one (1) displays the progress controls; a value of zero (0) does not display the progress controls.

The HasProgressControls method is used to optionally display progress controls on the report preview window. This method returns a BYTE representing the value of *hasprogresscontrol*. The Progress controls display the progress of the report when it is running. It displays records read, records selected, etc.).

Return Data Type: BYTE

Example:

```
oCrystal8.HasProgressControls(1)
```

HasRefreshButton (display refresh button on report preview)

HasRefreshButton (| *hasrefreshbutton* |)

HasRefreshButton Allows a refresh button on the report preview window.

hasrefreshbutton An integer constant, variable, EQUATE, or expression that specifies whether a refresh button will appear on the report's preview window. A value of one (1) displays the refresh button; a value of zero (0) does not display the refresh button.

The HasRefreshButton method is used to optionally display a refresh button on the report preview window. This method returns a BYTE representing the value of *hasrefreshbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasRefreshButton(1)
```

HasSearchButton (display search button on report preview)

HasSearchButton (| *hassearchbutton* |)

HasSearchButton Allows a search button on the report preview window.

hassearchbutton An integer constant, variable, EQUATE, or expression that specifies whether a search button will appear on the reports preview window. A value of one (1) displays the search button; a value of zero (0) does not display the search button.

The HasSearchButton method is used to optionally display a search button on the report preview window. This method returns a BYTE representing the value of *hassearchbutton*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasSearchButton(1)
```

HasZoomControl (display zoom control on report preview)

HasZoomControl (| *haszoomcontrol* |)

HasZoomControl Allows a zoom control on the report preview window.

haszoomcontrol An integer constant, variable, EQUATE, or expression that specifies whether a zoom button will appear on the report's preview window. A value of one (1) displays the zoom control; a value of zero (0) does not display the zoom control.

The HasZoomControl method is used to optionally display a zoom control on the report preview window. This method returns a BYTE representing the value of *haszoomcontrol*.

Return Data Type: BYTE

Example:

```
oCrystal8.HasZoomControl(1)
```

Init (initialize Crystal8 object)

Init (*reportname*)

Init Initialize the Crystal8 object.

reportname A string constant, variable, EQUATE, or expression containing the report name. This report name is used when previewing a report. It is also the window caption (text).

The Init method initializes the Crystal8 object. This method also sets the title of the preview window, if the report is previewed. A BYTE is returned from this method and represents whether the report engine is successfully initialized.

Return Data Type: BYTE

Example:

```
oCrystal8.Init( ReportPathName )
```

Kill (shut down Crystal8 object)

Kill

The Kill method shuts down the Crystal8 object, releasing any memory allocated durring the lifetime of the object.

Example:

```
oCrystal8.Kill()
```

Preview (preview a Crystal Report)

Preview (| *windowtitle*, *initstate*, *frame*, *icon*, *systemmenu*, *maximizebox*, *3dflag*|)

Preview Preview a Crystal Report.

windowtitle A string constant, variable, EQUATE, or expression containing the text to display in the report preview window's title bar. This parameter is optional.

initstate A string constant, variable, EQUATE, or expression containing an **N**, **M**, or **I**. Use **N** (Normal) to display the preview window at the default size. Use **M** (Maximized) to display the preview window in a maximized state. Use **I** (Iconized) to display the preview window in an iconized state. This parameter is optional.

frame A string constant, variable, EQUATE, or expression containing an **S**, **D**, **R**, or **N**. Use **S** to give the preview window a single pixel frame. Use **D** to give the preview window a thick frame. Use **R** to give the preview window a thick but resizeable frame. Use **N** to give the preview window a single pixel frame. This parameter is optional.

icon A string constant, variable, EQUATE, or expression containing an icon filename. By specifiying an icon file, a minimize button is automatically placed on the preview window. This parameter is optional.

systemmenu An integer constant, variable, EQUATE, or expression that specifies whether the preview window will contain a system menu. A value of TRUE will give the preview window a system menu. A value of FALSE (the default value) will not include the system menu on the preview window. This parameter is optional.

maximizebox An integer constant, variable, EQUATE, or expression that specifies whether the preview window will contain a maximize button. A value of TRUE (the default value) will place the maximize button on the preview window. A value of FALSE will not include the maximize box on the preview window. This parameter is optional.

3dflag An integer constant, variable, EQUATE, or expression that specifies whether the preview window will have a 3D look. The 3D look provides the window with a gray backgound and chiseled control look. A value of TRUE (the default value) will provide the preview window with the 3D look. A value of FALSE will not provide the preview window with the 3d look. This parameter is optional.

The Preview method is used to preview a Crystal report within a Clarion window. This method supports several preview window options.

Example:

```
oCrystal8.Preview( 'My Report','I','R',,0,1,1 )
```

_Print (print a Crystal Report)

_Print(| *copies*, *printersetup* |)

_Print Print a Crystal Report.

copies An integer constant, variable, EQUATE, or expression that specifies the number of copies of the report to print. The default for this parameter is 1. This parameter is optional.

printersetup An integer constant, variable, EQUATE, or expression that specifies whether the Printer Setup dialog is displayed before sending the report to the printer. Specifying TRUE or 1 for this parameter will cause the Printer Setup dialog to be displayed; a value of FALSE or 0 (the default value) will allow the report to go directly to the printer. This parameter is optional.

The _Print method prints a Crystal report directly to the printer without any option to preview the report. The printer setup dialog is optional before the report is sent to the printer.

Example:

```
oCrystal8._Print( 1, 1 )
```

Query (retrieve or set the SQL data query)

Query(| *querystring* |)

Query Set or retrieve the SQL data query.

querystring A string constant, variable, EQUATE, or expression containing the SQL query to be sent to the SQL data source. This parameter is optional.

The Query method is used to either get or set the SQL query. If the *querystring* is omitted from the method call, the current query is retrieved.

Return Data Type: STRING

Example:

```
formula = oCrystal8.Query()!retrieve query into formula variable
formula = '{{Customer.Country} in ["Australia"]' ! SQL query
oCrystal8.Query( formula ) ! Set the query
```

SelectionFormula (retrieve or set the Crystal formula)

SelectionFormula(| *formulastring* |)

SelectionFormula Set or retrieve the Crystal formula.

Formulastring A string constant, variable, EQUATE, or expression containing the Crystal formula. This parameter is optional.

The SelectionFormula method is used to either get or set the report's formula used to limit retrieved records. If the *formulastring* is omitted from the method call, the current formula is retrieved.

Return Data Type: STRING

Example:

```
formula = oCrystal8.SelectionFormula()!retrieve selection formula into formula variable
formula = '{{Customer.Country} in ["Australia"]' ! SQL query
oCrystal8.SelectionFormula( formula ) ! Set the query
```

ShowDocumentTips (show tips on docuement in the preview window)

ShowDocumentTips(| *showdocumenttips* |)

ShowDocumentTips Display tooltips in the document being previewed.

showdocumenttips An integer constant, variable, EQUATE, or expression that specifies whether to enable tooltips on the document in the report preview window. A value of one (1) indicates that tooltips will be shown. A value of zero (0) indicates that tooltips will not be displayed on the document in the preview window. This is the default if the parameter is omitted.

The ShowDocumentTips method is used to enable tooltips on the document being previewed in the report preview window. This method returns a BYTE representing the value of showdocumenttips.

Return Data Type: BYTE

Example:

```
oCrystal8.ShowDocumentTips(1)
```

ShowReportControls (show print controls)

ShowReportControls (| *showreportcontrols* |)

ShowReportControls Display tooltips in the document being previewed.

showreportcontrols An integer constant, variable, EQUATE, or expression that specifies whether the print controls are displayed. A value of one (1) will cause the print controls to be displayed. A value of zero (0) indicates that will hide the print controls. This parameter is optional and defaults to TRUE if omitted.

The **ShowReportControls** method is used to display the print controls. The print controls include the First, Previous, Next, and Last Page buttons as well as the buttons for Cancel, Close, Export, and Print to Printer. This method returns a BYTE representing the value of *showreportcontrols*.

Return Data Type: BYTE

Example:

```
oCrystal8.ShowReportControls(1)
```

ShowToolbarTips (show tips on preview window toolbar)

ShowToolbarTips(| *showtooltips* |)

ShowToolbarTips Display tooltips in the preview window's toolbar.

showtooltips An integer constant, variable, EQUATE, or expression that specifies whether to enable tooltips on the toolbar in the report preview window. A value of one (1) indicates that tooltips will be shown. This is the default if the parameter is omitted. A value of zero (0) indicates that tooltips will not be displayed in the report preview window.

The ShowToolbarTips method is used to enable tooltips on the toolbar of the report preview window. This method returns a BYTE representing the value of *showtooltips*.

Return Data Type: BYTE

Example:

```
oCrystal8.ShowToolBarTips(1)
```

Finance Library

AMORTIZE (amortize loan for specific number of payments)

APR (annual percentage rate)

COMPINT (compound interest)

CONTINT (continuous compounding interest)

DAYS360 (days difference based on 360-day year)

FV (future value)

IRR (internal rate of return)

NPV (net present value)

PERS (periods of annuity)

PREPERS (periods of annuity with prepayment)

PMT (payment of annuity)

PREFV (future value with prepayment)

PREPMT (payment of annuity with prepayment)

PREPV (present value with prepayment)

PV (present value)

RATE (rate of annuity)

PRERATE (rate of annuity with prepayment)

SIMPINT (simple interest)

AMORTIZE (amortize loan for specific number of payments)

AMORTIZE(*balance,rate,payment,totalpayments,principal,interest,endbalance*)

AMORTIZE	Calculates principal, interest, and remaining balance for a payment or payments.
balance	A numeric constant or variable containing the loan balance.
rate	A numeric constant or variable containing the *periodic interest rate* applied for a single period.
payment	A numeric constant or variable containing the desired payment (a negative number).
totalpayments	A numeric constant or variable containing the number of payments to amortize.
principal	The label of a DECIMAL variable to receive the portion of the payment(s) applied to pay back the loan (a negative number).
interest	The label of a DECIMAL variable to receive the portion of the payment(s) applied towards loan interest (a negative number).
endbalance	The label of a DECIMAL variable to receive the remaining loan balance.

The **AMORTIZE** procedure shows precisely which portion of a loan payment, or payments, constitutes interest and which portion constitutes repayment of the principal amount borrowed. The computed amounts are based upon a loan balance (*balance*), a periodic interest rate (*rate*), the payment amount (*payment*) and the number of payments (*totalpayments*). The remaining balance (*endbalance*) is also calculated.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Note:

The return parameters principal, interest, and endbalance must be DECIMAL values (passed by value).

Internal Formulas:

```
PRINCIPAL = payment + (balance * rate)
INTEREST = payment - principal
ENDINGBALANCE = balance + principal
```

Example:

```
Principal     DECIMAL(18,2)
Interest      DECIMAL(18,2)
EndingBalance DECIMAL(18,2)

CODE

BeginningBalance = LoanAmount             !Set first beginning balance
Period# = 1                               !Begin with the first period
LOOP Ptr# = Period# TO TotalPeriods       !Loop through the periods
 AMORTIZE(BeginningBalance,MonthlyRate,Payment,1, |
 Principal,InterestAmount,EndingBalance)  !Amortize 1 payment
 Q:Balance = BeginningBalance             !Show the beginning balance
 Q:Payment = Payment * (-1)               !..the payment amount
 Q:Principal = Principal * (-1)           !..amount applied to principal
 Q:Interest = InterestAmount * (-1)       !..amount applied to interest
 Q:NewBalance = EndingBalance             !...ending balance
 IF Ptr# = TotalPeriods|                  !If last period
  AND EndingBalance < 0                   !and balance went negative
  Q:Principal += EndingBalance            !adjust principal downard
  Q:Payment += EndingBalance              !and also the payment
  Q:NewBalance = 0.0                      !and make the balance zero.
 END
 EndingBalance = Q:NewBalance             !Save the ending balance
 ADD(AmortizeQue)                         !Add all period values to Q
 BeginningBalance = EndingBalance         !Make a new beginning balance
 TotalInterest += Q:Interest              !Add up total interest
END                                       !End loop
```

APR (annual percentage rate)

APR(*rate,periods*)

APR Returns the effective annual interest rate.

rate A numeric constant or variable containing the *contracted* interest rate.

periods A numeric constant or variable containing the number of compounding periods per year.

The **APR** function determines the effective annual rate of interest based upon the contracted interest rate (*rate*) and the number of compounding periods (*periods*) per year. For example, periods = 2 results in semi-annual compounding. The contracted interest rate is a "non-compounded annual interest rate."

Return DataType: DECIMAL

Internal Formulas:

$$APR = (1 + \frac{rate}{periods})^{periods} - 1$$

Example:

```
PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100) ! Setup Variables
RealRate = InterestRate / 100
AnnualRate = APR(RealRate,PeriodsPerYear) ! Call APR
AnnualRate *= 100                         ! Normalize Results
```

COMPINT (compound interest)

COMPINT(*principal,rate,periods*)

COMPINT Computes total compounded interest plus principal.

principal A numeric constant or variable containing the beginning balance, initial deposit, or loan.

rate A numeric constant or variable containing the *applied interest rate* for the given time frame.

periods A numeric constant or variable containing the number of compounding periods per year.

The **COMPINT** function computes total interest based on a principal amount (*principal*), an applied interest rate (*rate*), plus the number of compounding periods (*periods*). The computed amount includes the original principal plus the compound interest earned. *Periods* specifies the number of compounding periods. For example, periods = 2 results in semi-annual compounding.

Applied interest rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

AppliedRate = PeriodicRate * TotalPeriods

Return Data Type: DECIMAL

Internal Formulas:

$$\text{COMPOUND INTEREST} = \text{Principal} * \left(1 + \frac{\text{rate}}{\text{periods}}\right)^{\text{periods}}$$

Example:

```
CASE FIELD()
OF ?OK
 CASE EVENT()
 OF EVENT:Accepted
  PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)  ! Setup Variables
  ActualRate  = PeriodicRate * TotalPeriods
  CompoundInterest = COMPINT(Principal,ActualRate,TotalPeriods) ! Call COMPINT
  DO SyncWindow
```

```
END
```

CONTINT (continuous compounding interest)

CONTINT(*principal*,*rate*)

CONTINT Computes total continuously compounded interest.

principal A numeric constant or variable containing the beginning balance, initial deposit, or loan.

rate A numeric constant or variable containing the *applied interest rate* for the given time frame.

CONTINT computes total continuously compounded interest based on a principal amount (*principal*) and an applied interest rate (*rate*). The returned amount includes the original principal plus the interest earned. Applied interest rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

AppliedRate = PeriodicRate * TotalPeriods

Return Data Type: DECIMAL

Internal Formulas:

$$\text{CONTINUOUS COMPOUND INTEREST} = \text{Principal} * e^{\text{rate}}$$

Example:

```
PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100) ! Setup Variables
ActualRate  = PeriodicRate * TotalPeriods
ContinuousInterestAmount = CONTINT(Principal,ActualRate)   ! Call CONTINT
```

DAYS360 (days difference based on 360-day year)

DAYS360(*startdate*,*enddate*)

DAYS360 Computes the difference in days, between two given dates.

startdate A numeric constant or variable containing the beginning date.

enddate A numeric constant or variable containing the ending date.

DAYS360 determines the number of days difference between a beginning date (*startdate*) and an ending date (*enddate*), based on a 360-day year (30 day month). Both date parameters MUST contain Clarion standard date values.

Return Data Type: LONG

Internal Formulas:

DAYS DIFFERENCE = ending date - beginning date

where:

ending date = 360(year) + 30(month) + z

z = 30 if ending date = 31 and beginning date > 29

z = ending date if ending date <> 31 and beginning date < 29

and:

beginning date = 360(year) + 30(month) + z

z = 30 if beginning date = 31

z = beginning date if beginning date <> 31

Example:

```
DaysDifference = DAYS360(StartDate,EndDate)
```

FV (future value)

FV(*presentvalue*,*periods*,*rate*,*payment*)

FV	Computes the future value of an investment plus an income stream.
presentvalue	A numeric constant or variable containing the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
rate	A numeric constant or variable containing the *periodic* interest rate.
payment	A numeric constant or variable containing the periodic payment.

FV and **PREFV** determine the future value of an initial amount (*presentvalue*) plus an income stream. The income stream is defined as the total number of periods (*periods*), the periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the beginning of each period, use the PREFV function, which takes into account the added interest earned on each period's payment.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Return Data Type: DECIMAL

Internal Formulas:

$$FV = PresentValue(1 + rate)^{periods} + payment \frac{(1 + rate)^{int(periods)} - 1}{rate}$$

where int(periods) is the integer portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 FutureValue = PREFV(PresentValue,TotalPeriods,PeriodicRate,Payment)
ELSE
 FutureValue = FV(PresentValue,TotalPeriods,PeriodicRate,Payment)
END
```

IRR (internal rate of return)

IRR(*investment,cashflows,periods,rate*)

IRR	Computes the internal rate of return on an investment.
investment	A numeric constant or variable containing the initial cost of the investment (a negative number).
cashflows[]	A single dimensioned DECIMAL array containing the amounts of money paid out and received during discrete accounting periods.
periods[]	A single dimensioned DECIMAL array containing period numbers identifying the discrete accounting periods in which cash flows occurred.
rate	A numeric constant or variable containing the desired *periodic* rate of return.

IRR determines the rate of return on an investment (*investment*). The result is computed by determining the rate that equates the present value of future cash flows to the cost of the initial investment. The *cashflows* parameter is an array of money paid out and received during the course of the investment. The *periods* parameter is an array which contains the period number in which a corresponding cash flow occurred. The desired periodic rate of return (*rate*) for the investment is also specified. *Investment* should contain a negative number (a cost).

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Note:

Cashflows and *periods* MUST both be DECIMAL arrays of single dimension and equal size. The last element in the *periods* array MUST contain a zero to terminate the IRR analysis.

Return Data Type: DECIMAL

Internal Formulas:

$$0 = \frac{\text{cash flows}[1]}{(1 + \text{rate})^{\text{periods}[1]}} + \frac{\text{cash flows}[2]}{(1 + \text{rate})^{\text{periods}[2]}} + \ldots + \frac{\text{cash flows}[n]}{(1 + \text{rate})^{\text{periods}[n]}} + \text{investment}$$

The IRR function performs binary search iterations to home in on the return value. If more than 50 such iterations are required, a value of zero is returned.

Example:

```
CashFlows           DECIMAL(18,2),DIM(1000)
CashFlowPeriods     DECIMAL(4),DIM(1000)
InvestmentAmount    DECIMAL(18,2)
```

```
DesiredInterestRate DECIMAL(31,15)
Return              DECIMAL(10,6)

InvestmentTransactions FILE,DRIVER('TOPSPEED'),|
           NAME('busmath\!InvestmentTransactions'),PRE(INV),CREATE,THREAD
KeyIdPeriodNumber KEY(INVTRN:Id,INVTRN:PeriodNumber),NOCASE,PRIMARY
Notes             MEMO(1024)
Record            RECORD,PRE()
Id                 DECIMAL(8)
PeriodNumber       DECIMAL(8)
Date               ULONG
Type               STRING(20)
Amount             DECIMAL(16,2)
                  END
                       END

CODE

  CLEAR(CashFlows)                 !initialize queue
  CLEAR(CashFlowPeriods)           !initialize queue
  CLEAR(InvestmentAmount)
  CLEAR(TRANSACTION:RECORD)        !initialize record buffer
  DesiredInterestRate = INV:NominalReturnRate / (100 * INV:PeriodsPerYear)
  transcnt# = 0
  TRANSACTION:Id = INV:Id          !prime record buffer
  TRANSACTION:PeriodNumber = 0
  SET(TRANSACTION:KeyIdPeriodNumber,TRANSACTION:KeyIdPeriodNumber) !position file
  LOOP                             !loop for all transactions
   NEXT(InvestmentTransactions)    !read next record
   IF ERRORCODE() OR TRANSACTION:Id NOT = INV:Id  !if no more transactions
    BREAK                          !break from loop
   ELSE                            !else process transaction
    transcnt# += 1
    CashFlows[transcnt#] = TRANSACTION:Amount
    CashFlowPeriods[transcnt#] = TRANSACTION:PeriodNumber
   END                         !endelse process transaction
  END                          !endloop all transactions
  Return = IRR(InvestmentAmount,CashFlows,CashFlowPeriods,DesiredInterestRate)
  Return *= (100 * INV:PeriodsPerYear)       !normalize IRR to percent
```

NPV (net present value)

NPV(*investment,cashflows,periods,rate*)

NPV	Computes net present value of an investment.
investment	A numeric constant or variable containing the initial cost of the investment (a negative number).
cashflows[]	A single dimensioned DECIMAL array containing the amounts of money paid out and received during discrete accounting periods.
periods[]	A single dimensioned DECIMAL array containing period numbers identifying the discrete accounting periods in which cash flows occurred.
Rate	A numeric constant or variable containing the desired *periodic* rate of return.

NPV determines the viability of an investment proposal by calculating the present value of future returns, discounted at the marginal cost of capital minus the cost of the investment (*investment*). The *cashflows* parameter is an array of money paid out and received during the course of the investment. The *periods* parameter is an array which contains the period number in which a corresponding cash flow occurred. The desired periodic rate of return (*rate*) for the investment is also specified. *Investment* should contain a negative number (a cost).

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Note:

Cashflows and *periods* MUST both be DECIMAL arrays of single dimension and equal size. The last element in the *periods* array MUST contain a zero to terminate the NPV analysis.

Return Data Type: DECIMAL

Internal Formulas:

$$NPV = \frac{cash\ flows[1]}{(1 + rate)^{periods[1]}} + \frac{cash\ flows[2]}{(1 + rate)^{periods[2]}} + \ldots + \frac{cash\ flows[n]}{(1 + rate)^{periods[n]}} + investment$$

Example:

```
CashFlows           DECIMAL(18,2),DIM(1000)
CashFlowPeriods     DECIMAL(4),DIM(1000)
InvestmentAmount    DECIMAL(18,2)
DesiredInterestRate DECIMAL(31,15)
NetValue            DECIMAL(18,2)
```

```
InvestmentTransactions FILE,DRIVER('TOPSPEED'),|
            NAME('busmath\!InvestmentTransactions'),PRE(INV),CREATE,THREAD
KeyIdPeriodNumber KEY(INVTRN:Id,INVTRN:PeriodNumber),NOCASE,PRIMARY
Notes             MEMO(1024)
Record            RECORD,PRE()
Id                 DECIMAL(8)
PeriodNumber       DECIMAL(8)
Date               ULONG
Type               STRING(20)
Amount             DECIMAL(16,2)
                  END
                       END


CODE


  CLEAR(CashFlows)                 !initialize queue
  CLEAR(CashFlowPeriods)           !initialize queue
  CLEAR(InvestmentAmount)
  CLEAR(TRANSACTION:RECORD)        !initialize record buffer
  DesiredInterestRate = INV:NominalReturnRate / (100 * INV:PeriodsPerYear)
  transcnt# = 0
  TRANSACTION:Id = INV:Id          !prime record buffer
  TRANSACTION:PeriodNumber = 0
  SET(TRANSACTION:KeyIdPeriodNumber,TRANSACTION:KeyIdPeriodNumber) !position file
  LOOP                                  !loop for all transactions
   NEXT(InvestmentTransactions)         !read next record
   IF ERRORCODE() OR TRANSACTION:Id NOT = INV:Id  !if no more transactions
    BREAK                                        !break from loop
   ELSE                                          !else process transaction
    transcnt# += 1
    CashFlows[transcnt#] = TRANSACTION:Amount
    CashFlowPeriods[transcnt#] = TRANSACTION:PeriodNumber
   END                          !endelse process transaction
  END                           !endloop all transactions
  NetValue = NPV(InvestmentAmount,CashFlows,CashFlowPeriods,DesiredInterestRate)
```

PERS (periods of annuity)

PERS(*presentvalue,rate,payment,futurevalue*)

PERS	Computes the number of periods required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the present value of the investment.
rate	A numeric constant or variable containing the *periodic* rate of return.
payment	A numeric constant or variable containing the periodic payment.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PERS determines the number of periods required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), a periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the beginning of each period, use the PREPERS function, which takes into account the added interest earned on each period's payment.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Note:

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$0 = presentvalue(1+rate)^{frac(periods)} + payment\ \frac{1-(1+rate)^{int(-periods)}}{rate} - futurevalue(1+rate)^{int(-periods)}$$

where frac(periods) is the fractional portion of the *periods* parameter and where int(periods) is the integer portion of the *periods* parameter.

If the *payment* parameter is omitted or 0, then

$$PERS = \frac{log(futurevalue/presentvalue)}{log(1 + rate)}$$

If the *futurevalue* parameter is omitted or 0, then

$$PERS = \frac{log(1 - (rate * \frac{presentvalue}{-payment}))}{log(1 + rate)}$$

If the *presentvalue* parameter is omitted or 0, then

$$PERS = \frac{log(rate * \frac{presentvalue}{payment})}{log(1 + rate)}$$

The PERS function performs binary search iterations to home in on the periods value. If more than 50 such iterations are required, a value of zero is returned.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 TotalPeriods = PREPERS(PresentValue,PeriodicRate,Payment,FutureValue)
ELSE
 TotalPeriods = PERS(PresentValue,PeriodicRate,Payment,FutureValue)
END
```

PMT (payment of annuity)

PMT(*presentvalue,periods,rate,futurevalue*)

PMT	Computes the payment required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the amount of the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a payment is made.
Rate	A numeric constant or variable containing the *periodic* rate of return.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PMT determines the payment required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), a total number of periods (*periods*), and a periodic interest rate (*rate*). If payments occur at the beginning of each period then use the PREPMT function, which takes into account the added interest earned on each period's payment.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Note:

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$PMT = \frac{futurevalue(1+rate)^{int(-periods)} - presentvalue(1+rate)^{frac(periods)}}{\frac{1-(1+rate)^{int(-periods)}}{rate}}$$

where frac(periods) is the fractional portion of the *periods* parameter and
where int(periods) is the integer portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 Payment = PREPMT(PresentValue,TotalPeriods,PeriodicRate,FutureValue)
ELSE
 Payment = PMT(PresentValue,TotalPeriods,PeriodicRate,FutureValue)
END
```

PREPMT (payment of annuity with prepayment)

PREPMT(*presentvalue,periods,rate,futurevalue*)

PREPMT	Computes the payment required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the amount of the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a payment is made.
Rate	A numeric constant or variable containing the *periodic* rate of return.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PREPMT determines the payment required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), a total number of periods (*periods*), and a periodic interest rate (*rate*). If payments occur at the end of each period then use the PMT function, which calculates interest accordingly.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$PREPMT = \frac{futurevalue(1+rate)^{int(-periods)} - presentvalue(1+rate)^{frac(periods)}}{(1+rate)\,\frac{1-(1+rate)^{int(-periods)}}{rate}}$$

where frac(periods) is the fractional portion of the *periods* parameter and
where int(periods) is the integer portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 Payment = PREPMT(PresentValue,TotalPeriods,PeriodicRate,FutureValue)
ELSE
 Payment = PMT(PresentValue,TotalPeriods,PeriodicRate,FutureValue)
END
```

PREFV (future value with prepayment)

PREFV(*presentvalue*,*periods*,*rate*,*payment*)

PREFV	Computes the future value of an investment plus an income stream.
presentvalue	A numeric constant or variable containing the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
Rate	A numeric constant or variable containing the *periodic* interest rate.
payment	A numeric constant or variable containing the periodic payment.

FV and **PREFV** determine the future value of an initial amount (*presentvalue*) plus an income stream. The income stream is defined as the total number of periods (*periods*), the periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the beginning of each period then use the PREFV function, which takes into account the added interest earned on each period's payment.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

Return Data Type: DECIMAL

Internal Formulas:

$$PREFV = PresentValue(1 + rate)^{periods} + payment(1+rate)\frac{(1 + rate)^{int(periods)} - 1}{rate}$$

where int(periods) is the integer portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 FutureValue = PREFV(PresentValue,TotalPeriods,PeriodicRate,Payment)
ELSE
 FutureValue = FV(PresentValue,TotalPeriods,PeriodicRate,Payment)
END
```

PREPERS (periods of annuity with prepayment)

PREPERS(*presentvalue,rate,payment,futurevalue*)

PREPERS	Computes the number of periods required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the present value of the investment.
rate	A numeric constant or variable containing the *periodic* rate of return.
payment	A numeric constant or variable containing the periodic payment.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PREPERS determines the number of periods required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), a periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the end of each period, use the PERS function, which calculates interest accordingly.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$0 = presentvalue(1+rate)^{frac(periods)} + payment(1+rate)^{frac(periods)}\frac{1-(1+rate)^{int(-periods)}}{rate} - futurevalue(1+rate)^{int(-periods)}$$

where frac(periods) is the fractional portion of the *periods* parameter and where int(periods) is the integer portion of the *periods* parameter.

If the *payment* parameter is omitted or 0, then

$$PREPERS = \frac{log(futurevalue/presentvalue)}{log(1 + rate)}$$

If the *futurevalue* parameter is omitted or 0, then

$$PREPERS = \frac{log(1 - (\frac{rate}{(1+rate)} * \frac{presentvalue}{-payment}))}{log(1 + rate)}$$

If the *presentvalue* parameter is omitted or 0, then

$$PREPERS = \frac{log(\frac{rate}{(1+rate)}) * (\frac{futurevalue}{payment} + 1))}{log(1 + rate)}$$

The PREPERS function performs binary search iterations to home in on the periods value. If more than 50 such iterations are required, a value of zero is returned.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 TotalPeriods = PREPERS(PresentValue,PeriodicRate,Payment,FutureValue)
ELSE
 TotalPeriods = PERS(PresentValue,PeriodicRate,Payment,FutureValue)
END
```

PRERATE (rate of annuity with prepayment)

PRERATE(*presentvalue,periods,payment,futurevalue*)

PRERATE	Computes the periodic interest rate required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
payment	A numeric constant or variable containing the periodic payment amount.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PRERATE determines the periodic interest rate required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), the total number of periods (*periods*), and a payment amount (*payment*). If payments occur at the end of each period then use the RATE function, which calculates interest accordingly.

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$0=\text{presentvalue}(1+\text{rate})^{\text{frac(periods)}}+\text{payment}(1+\text{rate})^{\text{frac(periods)}}\frac{1-(1+\text{rate})^{\text{int(-periods)}}}{\text{rate}}-\text{futurevalue}(1+\text{rate})^{\text{int(-periods)}}$$

where frac(periods) is the fractional portion of the *periods* parameter and where int(periods) is the integer portion of the *periods* parameter.

If the *payment* parameter is omitted or 0, then

$$\text{PRERATE} = \left(\frac{\text{futurevalue}}{\text{presentvalue}}\right)^{1/\text{periods}} - 1$$

The PRERATE function performs binary search iterations to home in on the interest rate value. If more than 50 such iterations are required, RATE returns a value of zero.

Example:

```
IF TimeOfPayment = 'Beginning of Periods'
 AnnualRate = PRERATE(PresentValue,TotalPeriods,Payment,FutureValue)
 AnnualRate *= (PeriodsPerYear * 100)
ELSE
 AnnualRate = RATE(PresentValue,TotalPeriods,Payment,FutureValue)
 AnnualRate *= (PeriodsPerYear * 100)
END
```

PREPV (present value with prepayment)

PREPV(*periods,rate,payment,futurevalue*)

PREPV	Computes the present value required to reach a targeted future value.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
rate	A numeric constant or variable containing the *periodic* rate of return.
payment	A numeric constant or variable containing the periodic payment amount.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PREPV determines the present value required today to reach a desired amount (*futurevalue*) based upon the total number of periods (*periods*), a periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the end of each period then use the **PV** function, which calculates interest accordingly.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$PREPV = futurevalue(1+rate)^{-periods} - payment(1+rate)^{frac(-periods)} \frac{(1+rate)^{-periods} - (1+rate)}{rate}$$

where frac(periods) is the fractional portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 PresentValue = PREPV(TotalPeriods,PeriodicRate,Payment,FutureValue)
ELSE
 PresentValue = PV(TotalPeriods,PeriodicRate,Payment,FutureValue)
END
```

PV (present value)

PV(*periods,rate,payment,futurevalue*)

PV	Computes the present value required to reach a targeted future value.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
rate	A numeric constant or variable containing the *periodic* rate of return.
payment	A numeric constant or variable containing the periodic payment amount.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

PV determines the present value required today to reach a desired amount (*futurevalue*) based upon the total number of periods (*periods*), a periodic interest rate (*rate*), and a payment amount (*payment*). If payments occur at the beginning of each period then use the **PREPV** function, which takes into account the added interest earned on each period's payment.

Periodic rate may be calculated as follows:

PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100)

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$PV = futurevalue(1+rate)^{-periods} - payment\frac{(1+rate)^{frac(-periods)} - (1+rate)^{-periods}}{rate}$$

where frac(periods) is the fractional portion of the *periods* parameter.

Example:

```
PeriodicRate = AnnualRate / (PeriodsPerYear * 100)
IF TimeOfPayment = 'Beginning of Periods'
 PresentValue = PREPV(TotalPeriods,PeriodicRate,Payment,FutureValue)
ELSE
 PresentValue = PV(TotalPeriods,PeriodicRate,Payment,FutureValue)
END
```

RATE (rate of annuity)

RATE(*presentvalue,periods,payment,futurevalue*)

RATE	Computes the periodic interest rate required to reach a targeted future value.
presentvalue	A numeric constant or variable containing the present value of the investment.
periods	A numeric constant or variable containing the number of periods in which a cash flow occurred.
payment	A numeric constant or variable containing the periodic payment amount.
futurevalue	A numeric constant or variable containing the amount of the desired or targeted future value of the investment.

RATE determines the periodic interest rate required to reach a desired amount (*futurevalue*) based upon a starting amount (*presentvalue*), the total number of periods (*periods*), and a payment amount (*payment*). If payments occur at the beginning of each period then use the PRERATE function, which takes into account the added interest earned on each period's payment.

If the present value is less than the future value (annuities), payments are positive, and conversely, if the present value is greater than the future value (loans), payments are negative.

Return Data Type: DECIMAL

Internal Formulas:

$$0=presentvalue(1+rate)^{frac(periods)} + payment\ \frac{1-(1-rate)^{int(-periods)}}{rate} - futurevalue(1+rate)^{int(-periods)}$$

where frac(periods) is the fractional portion of the *periods* parameter and where int(periods) is the integer portion of the *periods* parameter.

If the *payment* parameter is omitted or 0, then

$$RATE = \left(\frac{futurevalue}{presentvalue}\right)^{1/periods} - 1$$

The RATE function performs binary search iterations to home in on the interest rate value. If more than 50 such iterations are required, RATE returns a value of zero.

Example:

```
IF TimeOfPayment = 'Beginning of Periods'
 AnnualRate = PRERATE(PresentValue,TotalPeriods,Payment,FutureValue)
 AnnualRate *= (PeriodsPerYear * 100)
ELSE
 AnnualRate = RATE(PresentValue,TotalPeriods,Payment,FutureValue)
 AnnualRate *= (PeriodsPerYear * 100)
END
```

SIMPINT (simple interest)

SIMPINT(*principal,rate*)

SIMPINT Returns simple (uncompounded) interest.

principal A numeric constant or variable containing the beginning balance, initial deposit, or loan.

rate A numeric constant or variable containing the simple or contract interest rate.

SIMPINT determines an interest amount based solely on a given amount (*principal*) and the simple interest rate (*rate*). The amount returned ONLY reflects interest earned.

Return Data Type: DECIMAL

Internal Formulas:

SIMPLE INTEREST = principal * rate

Example:

```
PeriodicRate = AnnualInterestRate / (PeriodsPerYear * 100) !Set up variables
ActualRate = PeriodicRate * TotalPeriods
SimpleInterestAmount = SIMPINT(Principal,ActualRate)    !Call SIMPINT
SimpleInterestAmount += Principal              !Add in principal
```

HTML Help Templates

HTML Help - Overview

HTML Help has emerged as the new standard help file format. HTML Help is distributed in a single (.chm) file. This file is compressed and made from several .html files. All images, table of contents, index, search files are compiled into this single .chm file. This makes for easier distribution of your help system.

HTML Help is available on any 32 bit windows platform, Windows 95/98, Windows 2000, NT 4, XP, and Vista. On Windows 95 and NT 4, the HTML Help Display Engine, Internet Explorer 4.0 or later, or Microsoft Office 2000 must be installed on the user's system.

Clarion's HTML Help implementation is a DLL that communicates with the Microsoft HTML System. The DLL is accessed by a Class Interface and is hooked to your application using simple standard Clarion code. Clarion's HTML Help intercepts the call to the Windows Help system when the F1 key is pressed. These calls are re-directed to the HTML Help System. There are several templates available which make the HTML Help system easily accessible from your Clarion program. Accessing Topics, Table of Contents, Indexing, Searching, Keywords, and Hyperlinks are a snap.

Clarion's HTML Help implementation is compatible with both the ABC and Legacy templates. It can only be used in a 32-bit application. Multiple HTML Help files may be called from a single application.

At the time of this writing, the Microsoft web site has extensive online documentation regarding design tips and considerations for a standard HTML help project. For example, it it strongly recommended that each of your windows has a specific topic, the standard OK, Cancel, Save button do not need specific help, etc.

HTML Help Example

HTML Help Templates

HTML Help Templates

cwHHGlobal extension template

cwHHProc extension template

AlinkLookUp code template

CloseHelp code template

GetHelpFileName code template

GetTopicName code template

KeyWordLookUp code template

SetHelpFileName code template

SetTopicName code template

ShowIndex code template

ShowSearch code template

ShowTOC code template

ShowTopic code template

cwHHGlobal extension template

The cwHHGlobal extension template is a requirement for any application that will utilize Clarion's HTML Help support. This extension simply includes the necessary equates and Classes needed to compile the application and defines the applications default help file.

Requirements

This global extension template has no template requirements.

Populating the Template

1. From the *Application Tree*, select the **Global Extensions** tab.
2. **Press** the INSERT button and select the cwHHGlobal extension.

Note: Depending on the Application Template your application uses (Clarion or ABC) the cwHHGlobal extension will be available beneath the Class cwHH – HTML Help for Clarion or Class cwHHABC – HTML Help for Clarion.

Template Prompts

The cwHHGlobal template extension provides the following prompts:

Default Help File Name
Specify the name of the application's compiled help (.chm) file. The filename should not be enclosed in quotes. A variable name can be used, but you must prepend an exclamation point to the variable name (i.e., *!HelpFileName*)

Append .HTM to Help Ids
Check this box to indicate that existing help IDs defined in the application will have a .HTM extension appended to them. This is useful when converting applications that use standard help ids to the HTML help standard.

Note:

This check box also makes it possible to have a WinHelp and HTML Help project maintained by the same application. The HTML Help template will strip out the leading tilde (~) from a WinHelp ID.

There is also a Utility template that will generate all HTML Help Ids to an output text file. See: ListHLPIds Utility Template.

Allow STD:Help on buttons to call HTML Help
Check this box if you wish to redirect the standard call for all buttons that use the STD:Help STD ID from a WinHelp (.HLP) file to the HTML help file named by the template. Clear this box if you need to have a dual-help application.

Allow STD:HelpIndex to call HTML Help Index
Check this box if you wish to redirect the standard call on any menu item or button that use the STD:HelpIndex STD ID from a WinHelp (.HLP) file to the Index of the HTML help file named by the template.

Allow STD:HelpSearch on buttons to call HTML Help Search
Check this box if you wish to redirect the standard call on any menu item or button that uses the STD:HelpSearch STD ID from a WinHelp (.HLP) file to the Search tab of the HTML help file named by the template.

Allow STD:HelponHelp on buttons to call HTML Help on…
Check this box if you wish to redirect the standard call on any menu item or button that uses the STD:HelponHelp STD ID from a WinHelp (.HLP) file to the Help on… section of the HTML help file named by the template.

Alternate 'TRAP' Key
Ordinarily the F12 (the trap key) or the F1 key initiates the call to an HTML help file. The trap key can be changed by entering the equate for the new keystroke to initiate the call to the help file. This allows developers who use the F12 for other reasons can continue to do so. You can also override this global trap key on the procedure level.

cwHHProc extension template

The cwHHProc extension template must be put on every procedure that will call the help file. When you add the cwHHGlobal extension, the template automatically adds the cwHHProc extension template to all appropriate (window based) procedures.

This template allows the help file to be different for each procedure, allows a specific topic page to be opened for the procedure and allows a customized keystroke to call the help file.

This template allows the help file to be different for each procedure, allows a specific topic page to be opened for the procedure and allows a customized keystroke to call the help file.

Requirements

This control template requires the cwHHGlobal global extension.

Populating the Template

Open the procedure that will contain the HTML Help extension.

1. **Select** the Extensions tab.

   The Extensions and Control Templates dialog displays.
2. **Press** the INSERT button and select the *cwHHProc* extension.

Template Prompts

The cwHHProc template extension provides the following prompts:

Override Help File name
Specifies the name of the procedure's compiled help (.chm) file. If a file is specified for the procedure, it will override the application's default help file. This may be left blank if there is no overriding help file.

Context URL
Specifies a compiled help (.chm) file or a topic within a help file that is to display when the user presses the F1 key. To specify a help topic use the 'SectionName/PageName.htm' naming convention. If the topic page is in the default section you simply use 'PageName.htm'. Remember that when using the Section/Page convention, the slash is a forward slash.

Alternate 'TRAP' Key
Ordinarily the F1 keystroke, and the F12 trap key (default) initiates the call to the help file. This value can be globally set in the cwHH Global Extension and can be changed here by entering the equate for the new keystroke to initiate the call to the help file.

AlinkLookUp code template

The AlinkLookUp code template is used to look up one or more Associative link (Alink) names within a compiled help (.chm) file. Associative links are used to link related help topics to each other. When a link that contains an Alink is clicked, a popup window appears with the list of related topics. This template calls the AlinkLookUp method. This method takes three parameters, list of Alinks, default message text and default window title.

Example method calls:

```
oHH.ALinkLookUp( sSearch, sMsg, sTitle ) !Find sSearch Alink
```

Template Prompts:

Alink to search for

Specify the Associative link(s) to search for. A string or variable may be specified. Multiple Alinks can be searched for by entering a list of Alinks separated by semicolons (;) as the string to search for. If a variable is used, the variable should contain a list of Alinks separated by semicolons. This entry is required.

String message to display if not found (opt)

Specifies the message to display in a message box if the Alink is not found. This entry is optional.

Title for Message box (opt)

Specifies the title of the message box dialog. This entry is optional.

CloseHelp code template

The CloseHelp code template closes any Help windows opened by the application. This template calls the CloseHelp method. There are no parameters for this method.

Example method calls:

```
oHH.CloseHelp() !Close all Help windows
```

Template Prompts:

There are no prompts for this code template.

GetHelpFileName code template

The GetHelpFileName code template retrieves the name of the compiled help (.chm) file. This template calls the GetHelpFile method. There are no parameters for this method. This method returns the Help file name.

Example method calls:

```
sHelpFileName = oHH.GetHelpFile( ) !Get Help file name
```

Template Prompts:

Variable to receive the Help File Name
Specifies a variable to be used to receive the HTML Help file name. Use the ellipsis (…) to select the variable from the field lists or type in a variable name that already exists.

GetTopicName code template

The GetTopicName code template gets the current help topic from the help object. This template calls the GetTopic method. This method does not take any parameters but returns the TopicName.

Example method calls:

```
sHelpFileName = oHH.GetTopic( ) !Get topic name
```

Template Prompts:

Variable to receive the Help Topic
Specifies a variable to receive the current help topic. Use the ellipsis (…) to select the variable from the field list or type in a variable name that already exists.

KeyWordLookUp code template

The KeyWordLookUp code template is used to look up one or more Keywords within a compiled help (.chm) file. Keywords are a collection of words and phrases that make up the help file's index. They are used find specific help topics. This template calls the KeywordLookUp method. This method takes three parameters, list of keywords, default message text and default window title.

Example method calls:

```
oHH.KeyWordLookUp( 'Demo' )
```

Template Prompts:

Key word to search for
Specify the Keywords(s) to search for. A string or variable may be specified. Multiple keywords can be searched for by entering a list of Keywords separated by semicolons (;) as the string to search for. If a variable is used, the variable should contain a list of keywords separated by semicolons. This entry is required.

String message to display if not found (opt)
Specifies the message to display in a message box if the Keyword is not found. This entry is optional.

Title for Message box (opt)
Specifies the title of the message box dialog. This entry is optional.

SetHelpFileName code template

The SetHelpFileName code template sets the name of the compiled help (.chm) file that the application will use. This template calls the SetHelpFile method. There is one parameter for this method. The parameter is the help file name.

Example method calls:

```
oHH.SetHelpFile( 'Demo.chm' ) !Set help file name for application
```

Template Prompts:

New HTML Help File Name
Specifies the help file to use for the application. This may be a string or a variable.

SetTopicName code template

The SetTopicName code template sets the current topic name. This should be set before displaying the topic. This template calls the SetTopic method. This method takes one parameter. It is the help topic name.

Example method calls:

```
oHH.SetTopic( 'Class_Interface/Class_Interface.htm' ) !Set topic name
```

Template Prompts:

New Topic
Specifies a topic page within the compiled help (.chm) file. To specify a help topic use the 'SectionName/PageName.htm' naming convention. If the topic page is in the default section you simply use 'PageName.htm'. Remember that when using the Section/Page convention, the slash is a forward slash.

ShowIndex code template

The ShowIndex code template opens the Index tab in the Navigation pane of the HTML Help Viewer and searches for the keyword, if specified. This template calls the ShowIndex method. This method takes one optional parameter, the keyword to search for.

Example method calls:

```
oHH.ShowIndex( ) !Opens Index tab
oHH.ShowIndex( 'Demo' ) !Opens Index tab and searches for keyword
```

Template Prompts:

Key word to search for (opt)
Specify a keyword to search for in the index. This entry is optional. If no keyword is specified, the Index tab is opened.

ShowSearch code template

The ShowSearch code template opens the Search tab in the Navigation pane of the HTML Help Viewer. This template calls the ShowSearch method. This method has no parameters.

Example method calls:

```
oHH.ShowSearch( )
```

Template Prompts:

There are no prompts for this code template.

ShowTOC code template

The ShowTOC code template selects the Contents tab in the Navigation pane of the HTML Help Viewer. This template calls the ShowTOC method. This method has no parameters.

Example method calls:

```
oHH.ShowTOC( )
```

Template Prompts:

There are no prompts for this code template.

ShowTopic code template

The ShowTopic code template opens the help topic specified by the SetTopicName code template (SetTopic method) in the HTML Help Viewer.

Example method calls:

```
oHH.ShowTopic( )
```

Template Prompts:

There are no prompts for this code template.

MenuStyleManager Class

The MenuStyle Manager class is designed to provide a simple interface to the XP Visual Styles currently available for standard menus.

MenuStyleManager Class Concepts

Many of the methods used here refer to menu coloring and gradients. A gradient is defined as a gradual progression of colors from a starting color to an ending color. Gradients can be applied in a horizontal or vertical direction over a given area.

There are also *Menu Parts* referenced within several of these methods. There are eight components of any MENUBAR structure, and each is described in the appropriate method where referenced.

Colors used in the MenuStyleManager are 32-bit, providing up to 4294967296 different color types.

Relationship to Other Application Builder Classes

The MenuStyleManager class works independently of all other ABC classes.

Template Implementation

The MenuStyleManager class is supported by the core templates in the Global Properties App Settings. This template instantiates an object based on a default *MenuStyleMgr* object name. The object is instantiated in the procedure where the application's main Frame exists.

MenuStyleManager Source Files

The MenuStyleManager class declarations are installed by default to the Clarion \LIBSRC folder. The MenuStyleManager source code and its respective components are contained in:

MenuStyle.INC	MenuStyleManager declarations
MenuStyle.CLW	MenuStyleManager method definitions
CWINT.INT	MenuStyleManager interface support

MenuStyleManager Properties

MenuFEQ(Menu Field Equate)

```
MenuFEQ     LONG,PROTECTED
```

The **MenuFEQ** property holds the active MENUBAR Field Equate Number.

Implementation:

The **MenuFEQ** property is assigned to the **MenuInterface** Interface in the **InitMenuInterface** method.

See Also: **InitMenuInterface(initialize menu properties)**

MenuStyleManager Methods

AlignShortcutsToLeft (position menu shortcuts to left)

AreShortcutsLeftAligned (detect left aligned menu shortcuts)

GetEndColor (get ending gradient color)

GetFlatMode (get menu flat mode)

GetFont(get menu text font)

GetImage (get image name assigned to menu part)

GetIsVerticalGradient (does menu part use a vertical gradient)

GetStartColor (get starting gradient color)

GetTextColor(get menu text color)

Init (initialize the MenuStyleManager object)

InitMenuInterface (initialize menu properties)

MenuHasGradient (does menu part have gradient applied)

MenuInterface (IMenuInterface reference)

MenuStyle (IMenuStyle reference)

MixColors (mix two colors)

SetColor (specify color for menu part)

SetFlatMode (set menu flat mode)

SetFont (set menu text font)

SetImage (apply image to menu part)

SetTextColor (set menu text color)

SetThemeColors (set colors based on theme detected)

SetVerticalGradient (set vertical color gradient)

AlignShortcutsToLeft (position menu shortcuts to left)

AlignShortcutsToLeft(*value* **)**

value An optional BYTE value used to position any menu shortcuts. The default value is 1.

The **AlignShortcutsToLeft** method is used to left align the menu shortcuts or hot keys specified. These shortcuts always appear to the right of the menu text. This method is used to simply determine how they are aligned. If *value* is zero (0), the shortcuts are right aligned.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the alignment of menu shortcuts at runtime.

See Also: **AreShortcutsLeftAligned (detect left aligned menu shortcuts)**

AreShortcutsLeftAligned (detect left aligned menu shortcuts)

AreShortcutsLeftAligned( ), BYTE

The **AreShortcutsLeftAligned** method is used to detect if the shortcut of any menu are left aligned, If the return value is TRUE (1), the menu shortcuts are left aligned. Otherwise, if the function returns FALSE (0), they are right aligned.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the alignment of menu shortcuts at runtime.

Return Data Type: BYTE

See Also: **AlignShortcutsToLeft (position menu shortcuts to left)**

GetEndColor (get ending gradient color)

GetEndColor(*MenuPart* **), LONG**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

The **GetEndColor** method is used to retrieve the ending color of any *MenuPart*. The ending color represents the finishing color of any menu gradient, and is represented by a LONG integer.

Implementation:
This method is used by the **SetVerticalGradient** method to determine the ending color of the specific vertical color gradient to apply.

Return Data Type: LONG

See Also: **SetVerticalGradient (set vertical color gradient)**

GetFlatMode(get menu flat mode)

GetFlatMode( ), BYTE

The **GetFlatMode** method is used to detect if a menu is currently using a flat mode appearance. If the method returns TRUE (1), flat mode is active.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the flat mode menu appearance at runtime.

Return Data Type: BYTE

See Also: **SetFlatMode(set menu flat mode)**

GetFont(get menu text font)

GetFont(**FontName*,**FontSize*,**FontStyle*,**FontCharSet*,**FontAngle*,**FontColor***)**

The **GetFont** method is used to return the font characteristics for the active menu bar. The returned values are assigned to a set of internally referenced values:

```
FontName     *CSTRING,RAW
FontSize     SIGNED
FontStyle    DWORD
FontCharSet  BYTE
FontAngle    LONG
FontColor    COLORREF
```

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu font at runtime.

See Also: **SetFont (set menu text font)**

GetImage (get image name assigned to menu part)

GetImage(*MenuPart* **), STRING**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

The **GetImage** method is used to return the image name of a specified *ManuPart*.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu appearance by specifying an image for a target menu area at runtime.

Return Data Type: STRING

See Also: **SetImage (apply image to menu part)**

GetIsVerticalGradient (does menu part use a vertical gradient)

GetIsVerticalGradient(*MenuPart* **), BYTE**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

The **GetIsVerticalGradient** method is used to detect that a target *MenuPart* is using a vertical gradient.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu appearance by specifying a vertical gradient at runtime.

Return Data Type: BYTE

See Also: **SetVerticalGradient (set vertical color gradient)**

GetStartColor (get starting gradient color)

GetStartColor(*MenuPart* **), LONG**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

The **GetStartColor** method is used to retrieve the starting color of any *MenuPart*. The starting color represents the initial color of any menu gradient, and is represented by a LONG integer.

Implementation:
This method is used by the **SetVerticalGradient** method to determine the starting color of the specific vertical gradient to apply.

Return Data Type: LONG

See Also: **SetVerticalGradient (set vertical color gradient)**

GetTextColor(get menu text color)

GetTextColor(*MenuTextType* **), LONG**

MenuTextType A BYTE value representing a specific type of menu text to color. There are a series of internal equates available to easily identify these elements:

MenuColors:NormalText	EQUATE(1)
MenuColors:SelectedText	EQUATE(2)
MenuColors:HotText	EQUATE(3)
MenuColors:InactiveText	EQUATE(4)

The **GetTextColor** method gets a color for the specified *MenuTextType*. It returns a LONG integer representing the 32-bit color used.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu text color at runtime.

Return Data Type: STRING

See Also: **SetTextColor(get menu text color)**

Init (initialize the MenuStyleManager object)

Init(*MenuFEQ* **)**

MenuFEQ A numeric constant, variable, EQUATE, or expression containing the control number of the filedrop's LIST control.

The **Init** method initializes the **MenuStyleManager** object.

Implementation:
The **Init** method is used to initialize the target **MenuStyleManager** object, and identifies the target MENUBAR to modify via the MENUBAR Field Equate Label.

Example:

```
  MenuStyleMgr.Init(?MenuBar)

  MenuStyleMgr.SetFlatMode(False)

  MenuStyleMgr.SetColor(MenuBrushes:ImageBkgnd,-2147483644,-2147483644)

  MenuStyleMgr.SetColor(MenuBrushes:SelectedBkgnd,8388608,8388608,False)

  MenuStyleMgr.SetColor(MenuBrushes:SelectedBarBkgnd,8388608,8388608,True)

  MenuStyleMgr.SetColor(MenuBrushes:HotBkgnd,8388608,8388608,True)

  MenuStyleMgr.SetColor(MenuBrushes:FrameBrush,8388608, 8388608,True)

  AppFrame{PROP:Text} = 'Visual Styles Demo -
Windows Classic Menu Style Active'

  DISPLAY()
```

See Also: **InitMenuInterface(initialize menu properties)**

InitMenuInterface (initialize menu properties)

InitMenuInterface()

The **InitMenuInterface** is a protected method that is used to initialize several menu components of the target menu used to apply a selected visual style. These components include the Menu Field Equate Label, Style, Flat Mode, Colors and Brush style.

Implementation:
The protected **InitMenuInterface** method is called by the Init method of the instantiated **MenuStyleManager** class, and is called prior to assigning any XP Menu Style.

See Also: **Init (initialize the MenuStyleManager object)**

MenuHasGradient (does menu part have gradient applied)

MenuHasGradient(*MenuPart***), BYTE**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

The **MenuHasGradient** method is used to detect if a gradient color is applied to a specified target *MenuPart*. If the method returns TRUE, a gradient color is currently in use.

Implementation:
This method is currently not implemented by the **MenuClassManager** object, but is available to the developer if manual processing of any MENUBAR is preferred.

Return Data Type: **BYTE**

See Also: **SetColor (specify color for menu part)**

MenuInterface (IMenuInterface reference)

MenuInterface &IMenuInterface

The MenuInterface is a PROTECTED reference to the IMenuInterface.declared in CWINI,INT as follows:

```
IMenuInterface INTERFACE,COM


GetStyle PROCEDURE (),*IMenuStyle
SetStyle PROCEDURE (<*IMenuStyle>)
END
```

Implementation:
The MenuInterface interface is implemented through the **InitMenuInterface** method, and is used to query the active menu style of the current application.

See Also: InitMenuInterface(initialize menu properties)

MenuStyle (IMenuStyle reference)

MenuStyle &IMenuStyle

The **MenuStyle** is a PROTECTED reference to the **IMenuStyle** interface declared in CWINI,INT as follows:

```
IMenuStyle      INTERFACE,COM
Clone             PROCEDURE (),*IMenuStyle
Destroy           PROCEDURE ()
SetDirty          PROCEDURE ()
MenuFont          PROCEDURE (),*IFontProperties
MenuBrush         PROCEDURE (UNSIGNED),*IBrush    !Index: MenuBrushes
MenuColor         PROCEDURE (UNSIGNED, COLORREF)  !Index: MenuColors
MenuColor         PROCEDURE (UNSIGNED),COLORREF   !Index: MenuColors
FlatMode          PROCEDURE (UNSIGNED)            !Value: FlatMenuMode
FlatMode          PROCEDURE (),UNSIGNED           !Value: FlatMenuMode
AlignShortcuts    PROCEDURE (BOOLEAN)
AlignShortcuts    PROCEDURE (),BOOLEAN  ! TRUE: Shortcuts are left aligned
                END
```

Implementation:
The MenuStyle interface is implemented through the **InitMenuInterface** method, and is used to hold the active menu style of the current application. The interface methods described above are used to modify various characteristics of the target menu structure.

See Also: InitMenuInterface(initialize menu properties)

MixColors (mix two colors)

MixColors(*Color1*, *Color2*, *Percentage* **),LONG**

Color1	A LONG value that represents a valid RGB composite color to be mixed with the second color (color1)
Color2	A LONG value that represents a valid RGB composite color to be mixed with the first color (color1)
percentage	A BYTE value that represents the percentage of mix applied to the two colors.

The MixColors method is a PROTECTED method used to calculate and return a composite mix of two colors with a given percentage of mix.

Each color passed to this method is a RGB composite color extracted as follows:

```
InputColor  LONG
RGBT1       GROUP,OVER(InputColor)
R            BYTE
G            BYTE
B            BYTE
NotUsed      BYTE
            END
```

The mixing of colors is applied using the following formula to all RGB elements:

```
(RGBT1.R + ((RGBT2.R - RGBT1.R) * pPercentageMix/100))
```

Return Value: **LONG**

Implementation:

The **MIxColors** method is used by the **SetThemeColors** method to detect and set the proper theme menu style color based on the active theme detected.

See Also: SetThemeColors (set colors based on theme detected)

SetColor (specify color for menu part)

SetColor(*MenuPart*, *StartColor*, *EndColor*,< *vertical* >**)**

MenuPart A BYTE value representing the area of the menu to color. There is a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

StartColor A LONG integer that specifies a starting color to use. If a gradient is active, this represents the starting color of the menu gradient.

EndColor A LONG integer that specifies a starting color to use. If a gradient is active, this represents the finishing color of the menu gradient.

vertical An optional BYTE value that specifies the direction of the color gradient. When TRUE (1), the direction of the gradient color is vertical. Otherwise, if omitted or False, the gradient is horizontal.

The **SetColor** method is used to set a specific color gradient to a specified part of the active menu. In general, you may modify any individual part of a menu as specified by the **MenuPart** parameter, however most styles are a group of menu elements assigned to produce a specific look. See the example below for an illustration.

Implementation:
The **SetColor** method can be applied anytime after the **Init** method for the target menu has been executed.

Example:

```
MenuStyleMgr.SetColor(MenuBrushes:SelectedBkgnd,15331781,16764861,False)
MenuStyleMgr.SetColor(MenuBrushes:SelectedBarBkgnd,15331781,16760187,True)
MenuStyleMgr.SetColor(MenuBrushes:HotBkgnd,15331781,16764861,True)
MenuStyleMgr.SetColor(MenuBrushes:FrameBrush,12615680, 12615680,True)
```

See Also: **Init (initialize the MenuStyleManager object)**

SetFlatMode (set menu flat mode)

SetFlatMode(*mode* **)**

mode A BYTE value that sets the menu flat mode. A value of TRUE (1) turns flat mode on, and a value of FALSE (0) turns it off.

The **SetFlatMode** method controls the flat mode appearance of a target menu. When flat mode is off (FALSE), the selection bar appears "raised" in the menu structure. Otherwise the appearance of the selection bar is "flat".

Implementation:
The method is usually called right after the **Init** method, and prior to setting menu colors.

Example:

```
MenuStyleMgr.Init(?MenuBar)

MenuStyleMgr.SetFlatMode(False)

MenuStyleMgr.SetColor(MenuBrushes:ImageBkgnd,12632256,12632256)

MenuStyleMgr.SetColor(MenuBrushes:SelectedBkgnd,12632256,12632256,False)

MenuStyleMgr.SetColor(MenuBrushes:SelectedBarBkgnd,12632256,12632256,True)

MenuStyleMgr.SetColor(MenuBrushes:HotBkgnd,12632256,12632256,True)

MenuStyleMgr.SetColor(MenuBrushes:FrameBrush,8421504, 8421504,True)

AppFrame{PROP:Text} = 'Visual Styles Demo - Window2K Menu Style Active'

DISPLAY()
```

See Also: **GetFlatMode(get menu flat mode)**

SetFont (set menu text font)

SetFont(*FontName* , *FontSize* , *FontStyle* , *FontCharSet* , *FontAngle* , *FontColor*)

FontName	A string constant or variable containing the name of the font.
FontSize	An integer constant or variable containing the size (in points) of the font. If omitted, the system default font size is used.
FontStyle	An integer constant, constant expression, EQUATE, or variable specifying the strike weight and style of the font. If omitted, the weight is normal.
FontCharSet	A LONG integer variable specifying the character set value
FontAngle	An integer constant or constant expression that specifies the amount of rotation, in tenths of degrees. If positive, the angle is measured counter-clockwise from the report's horizontal orientation. Valid values are between 3600 and -3600
FontColor	A LONG integer constant or variable containing the red, green, and blue values for the color of the font in the low-order three bytes, or an EQUATE for a standard Windows color value. If omitted, black is used.

The **SetFont** method is used to set the default font used for the active MENUBAR.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu font at runtime.

Return Data Type: STRING

See Also: **GetFont(get menu text font)**

SetImage (apply image to menu part)

SetImage (*MenuPart* **,** *ImageName* **)**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

ImageName A string constant or variable containing the name of the file to display.

The **SetImage** method is used to apply a valid image to a target *MenuPart*, This method is generally used to replace any color gradient assignment with a more complex color image of watermark.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu appearance by specifying an image for a target menu area at runtime.

See Also: GetImage (get image name assigned to menu part)

SetTextColor (set menu text color)

SetTextColor(*MenuTextType* **,** *TextColor* **)**

MenuTextType A BYTE value representing a specific type of menu text to color. There are a series of internal equates available to easily identify these elements:

MenuColors:NormalText	EQUATE(1)
MenuColors:SelectedText	EQUATE(2)
MenuColors:HotText	EQUATE(3)
MenuColors:InactiveText	EQUATE(4)

TextColor A LONG integer that specifies a text color to use. There are approximately 4294967296 different colors in the 32-bit library.

The **SetTextColor** method sets a color for a specified *MenuTextType*.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu text color at runtime.

See Also: **GetTextColor(get menu text color)**

SetThemeColors (set colors based on theme detected)

SetThemeColors( )

The **SetThemeColors** method is used to detect the active theme and set a Menu Style method to automatically match it.

Implementation:

The **SetThemeColors** method is not used by the **MenuStyleManager** class, but was created to allow the developer to easily apply an automatic Menu Style under program control.

If an active theme is not detected, the *Windows Classic* Menu Style is automatically applied.

Example:

```
OF ?MenuStylesAutomatic  !Set MenuStyle to match active theme
 MenuStyleMgr.Init(?MenuBar)
 MenuStyleMgr.SetThemeColors()
 AppFrame{PROP:Text} = 'Menu Styles Demo - Automatic'
```

SetVerticalGradient (set vertical color gradient)

SetVerticalGradient(*MenuPart* **,<***vertical***>)**

MenuPart A BYTE value representing a specific area of the menu. There are a series of internal equates available to easily identify these elements:

MenuBrushes:NormalBkgnd	EQUATE(1) !Normal Background
MenuBrushes:SelectedBkgnd	EQUATE(2) !Selected Background
MenuBrushes:HotBkgnd	EQUATE(3) !Hot Background
MenuBrushes:ImageBkgnd	EQUATE(4) !Image Background
MenuBrushes:NormalBarBkgnd	EQUATE(5) !Normal Select Bar Background
MenuBrushes:SelectedBarBkgnd	EQUATE(6) !Selected Bar Background
MenuBrushes:GrayBrush	EQUATE(7) !Gray Brush
MenuBrushes:FrameBrush	EQUATE(8) !Frame Brush

vertical An optional BYTE value that specifies the direction of the color gradient. When TRUE (1), the direction of the gradient color is vertical. Otherwise, if omitted or False, the gradient is horizontal.

The **SetVerticalGradient** method is used to designate that any colors applied to the target *MenuPart* will be in a vertical direction.

Implementation:
The method is not called anywhere within the **MenuStyleManager** Class. It is an optional method that can be used by the developer to detect and possibly change the menu appearance by specifying a vertical gradient at runtime. It should be called prior to the **SetColor** method.

Return Data Type: STRING

See Also: **GetStartColor (get starting gradient color)**, **GetEndColor (get ending gradient color)**, **SetColor (specify color for menu part)**

Statistics Library

FACTORIAL (factorial of a number)

FREQUENCY (frequency count of an item in a set)

LOWERQUARTILE (lower quartile value of a set)

MEAN (mean of a set)

MEDIAN (median of a set)

MIDRANGE (midrange of a set)

MODE (mode of a set)

PERCENTILE (pth percentile value of a set)

RANGEOFSET (range of a set)

RVALUE (linear regression correlation coefficient)

SS (sum of squares)

SSXY (sum of squares for x and y)

ST1 (student's t for a single mean)

SDEVIATIONP (standard deviation of a population)

SDEVIATIONS (standard deviation of a sample)

SUMM (summation of a set)

UPPERQUARTILE (upper quartile value of a set)

VARIANCEP (variance of a population)

VARIANCES (variance of a sample)

FACTORIAL (factorial of a number)

FACTORIAL(number)

FACTORIAL	Computes the factorial of a number.
number	A numeric constant or variable containing a positive integer value.

The FACTORIAL function implements the standard factorial formula. For example, if the number provided is 5 then the function returns the value of: (1 x 2 x 3 x 4 x 5) = 120.

Return DataType: REAL

Example:

```
X = 5
FactorialOfX = FACTORIAL(X)     !call FACTORIAL
```

FREQUENCY (frequency count of an item in a set)

FREQUENCY(*dataset,searchvalue*)

FREQUENCY	Counts the number of instances of a particular value within a numeric set.
dataset	The label of a QUEUE with its first component field defined as a REAL.
searchvalue	A numeric constant or variable containing the value to search for in the QUEUE.

FREQUENCY counts the number of instances of a particular value (*searchvalue*) within a numeric set (*dataset*). The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

For example, given the data set: [1,2,2,3,4,5] and the search value 2, the function would return a frequency count of 2.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX  QUEUE,PRE()
X          REAL
          END

FrequencyOfX REAL
SearchValue  REAL(1)

CODE
    FREE(StatSetX)                    !free the QUEUE
    CLEAR(STADAT:RECORD)              !clear the record buffer
    STADAT:Id = STA:Id                !prime the record buffer
    STADAT:ItemNumber = 1
    SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
    LOOP                              !load the QUEUE
      NEXT(StatisticsData)            !read next record
      IF ERRORCODE() OR STADAT:Id NOT = STA:Id
        BREAK
      ELSE
        StatSetX:X = STADAT:X         !load the QUEUE buffer
        ADD(StatSetX)                 !add to the QUEUE
      END
    END
   FrequencyOfX = FREQUENCY(StatSetX,SearchValue) !call FREQUENCY to search Q
```

LOWERQUARTILE (lower quartile value of a set)

LOWERQUARTILE(*dataset*)

LOWERQUARTILE	Returns the median of the lower half of an ordered numeric data set.
Dataset	The label of a QUEUE with its first component field defined as a REAL.

LOWERQUARTILE computes a value such that at most 25% of a numeric set's values are less than the computed value, and at most 75% of the set's values are greater than the computed value. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*). For example, given the data set: [1,2,3,4,5,6,7,8] the lower quartile value is 2.5.

In general, the LOWERQUARTILE function is only meaningful when the number of elements in the data set is large (ie. greater than 20).

See also:

PERCENTILE, UPPERQUARTILE.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX  QUEUE,PRE()
X          REAL
          END

LowerQuartileOfSet  REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X         !load the QUEUE buffer
     ADD(StatSetX)                 !add to the QUEUE
   END
 END
 LowerQuartileOfSet = LOWERQUARTILE(StatSetX)    !call LOWERQUARTILE
```

MEAN (mean of a set)

MEAN(dataset)

MEAN	Returns the mean or average of a set of numbers.
dataset	The label of a QUEUE with its first component field defined as a REAL.

MEAN computes the arithmetic average of a set. The arithmetic average is the sum of a set's values divided by the number of elements in the set. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (dataset).

For example, if the set contains 5 values: [1,2,3,4,5] then the mean is (1+2+3+4+5) / 5 = 3.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX  QUEUE,PRE()
X          REAL
          END

MeanOfSet   REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X         !load the QUEUE buffer
     ADD(StatSetX)                 !add to the QUEUE
   END
  END
 MeanOfSet = MEAN(StatSetX)        !call MEAN
```

MEDIAN (median of a set)

MEDIAN(*dataset*)

MEDIAN Returns the middle value of an ordered numeric data set.

dataset The label of a QUEUE with its first component field defined as a REAL.

MEDIAN finds the value which is exactly in the middle when all of the data is in (sorted) order from low to high, or the mean of the two data values which are nearest the middle. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

For example, given the data set: [1,2,3,4,5] the median is 3, or given the data set: [1,2,4,5] the median is (2 + 4) / 2 = 3.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX  QUEUE,PRE()
X       REAL
      END

MedianOfSet  REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X          !load the QUEUE buffer
     ADD(StatSetX)                  !add to the QUEUE
   END
 END
 MedianOfSet = MEDIAN(StatSetX)     !call MEDIAN
```

MIDRANGE (midrange of a set)

MIDRANGE(*dataset*)

MIDRANGE	Returns the average of the highest and lowest value in a numeric set.
dataset	The label of a QUEUE with its first component field defined as a REAL.

MIDRANGE computes the mean of the smallest and largest values in a data set. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

For example, given the data set: [1,2,3,4,5] the midrange is (1 + 5) / 2 = 3.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX QUEUE,PRE()
X         REAL
         END

MidrangeOfSet   REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
    BREAK
   ELSE
    StatSetX:X = STADAT:X          !load the QUEUE buffer
    ADD(StatSetX)                  !add to the QUEUE
   END
 END
 MidrangeOfSet = MIDRANGE(StatSetX) !call MIDRANGE
```

MODE (mode of a set)

MODE(*dataset*,*modeset*)

MODE	Identifies the value(s) that occurs most often in a numeric set and returns the number of times it occurs.
dataset	The label of a QUEUE with its first component field defined as a REAL.
modeset	The label of a QUEUE with its first component field defined as a REAL.

MODE determines which value or values within a numeric set occurs most often. The value or values are then stored in the passed QUEUE (*modeset*), and the function returns the number of occurrences (mode count).

The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*). *Modeset* must be a QUEUE with the first component defined as a REAL variable. The contents of the *modeset* QUEUE are freed upon entry to the function.

For example, given the data set: [5,5,7,7,9] the returned mode count is 2 and the *modeset* QUEUE would contain two separate entries: 5 and 7.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX   QUEUE,PRE()
X           REAL
           END
ModeSet    QUEUE,PRE()
ModeValue   REAL
           END
ModeCount  REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X         !load the QUEUE buffer
     ADD(StatSetX)                 !add to the QUEUE
   END
 END
 ModeCount = MODE(StatSetX,ModeSet) !call MODE
```

PERCENTILE (pth percentile value of a set)

PERCENTILE(dataset,percentile)

PERCENTILE	Returns a value that divides an ordered numeric data set into two portions of specified relative size.
dataset	The label of a QUEUE with its first component field defined as a REAL.
percentile	A numeric constant or variable containing an integer value in the range: 1 >= p <= 99 .

PERCENTILE computes a value such that at most pth% of the set's values are less than the amount, and at most 100 - pth% of the set's values are greater than the amount. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (dataset).

For example, given the data set: [1,2,3,4,5,6], the 50th Percentile value is 3.5.

In general, the PERCENTILE function is only meaningful when the number of elements in the data set is large (ie. greater than 20).

See also:

LOWERQUARTILE

UPPERQUARTILE.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX   QUEUE,PRE()
X           REAL
           END

PercentileOfX   REAL
DividePoint   REAL(90)

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X          !load the QUEUE buffer
     ADD(StatSetX)                  !add to the QUEUE
   END
 END
 PercentileOfX = PERCENTILE(StatSetX,DividePoint) !call PERCENTILE
```

RANGEOFSET (range of a set)

RANGEOFSET(*dataset*)

RANGEOFSET	Returns the difference between the largest and smallest values in a numeric set.
dataset	The label of a QUEUE with its first component field defined as a REAL.

RANGEOFSET computes the difference between the largest and smallest values in a numeric set. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

For example, given the data set: [1,2,3,4,5], the range is (5 - 1) = 4.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX   QUEUE,PRE()
X        REAL
       END

RangeOfSetX   REAL

CODE
 FREE(StatSetX)                    !free the QUEUE
 CLEAR(STADAT:RECORD)              !clear the record buffer
 STADAT:Id = STA:Id                !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                              !load the QUEUE
   NEXT(StatisticsData)            !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X         !load the QUEUE buffer
     ADD(StatSetX)                 !add to the QUEUE
   END
 END
 RangeOfSetX = RANGEOFSET(StatSetX) !call RANGEOFSET
```

RVALUE (linear regression correlation coefficient)

RVALUE(*dataset* [,*meanX*] [,*meanY*])

RVALUE	Returns the correlation coefficient of the linear relationship between the paired data points in the data set.
dataset	The label of a QUEUE with its first two component fields defined as REAL. The first component contains the set of X values and the second component contains the set of Y values.
meanX	A numeric constant or variable containing the average of the X values (optional).
meanY	A numeric constant or variable containing the average of the Y values (optional).

RVALUE computes the correlation coefficient of the linear relationship between the paired data points in the data set. The function operates on the numeric sets defined by all the entries in the first two components of the designated QUEUE (*dataset*).

The optional parameters help to optimize the function. If not provided, the value(s) are computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY QUEUE,PRE()
X          REAL
Y          REAL
          END
CorrelationCoefficient  REAL

CODE
 FREE(StatSetXY)                    !free the QUEUE
 CLEAR(STADAT:RECORD)               !clear the record buffer
 STADAT:Id = STA:Id                 !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                               !load the QUEUE
   NEXT(StatisticsData)             !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetXY:X = STADAT:X         !load the QUEUE buffer
     StatSetXY:Y = STADAT:Y         !load the QUEUE buffer
     ADD(StatSetXY)                 !add to the QUEUE
   END
 END
 CorrelationCoefficient = RVALUE(StatSetXY)    !call RVALUE
```

SS (sum of squares)

SS(*dataset* [,*meanofset*])

SS	Returns the sum of the squared differences between each data set value and the data set's mean.
dataset	The label of a QUEUE with its first component field defined as a REAL.
meanofset	A numeric constant or variable representing the average of the data set's values (optional).

SS computes the sum of the squared differences between each data set value and the data set's mean. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*). The computation is a significant factor in several statistical formulas.

The optional parameter helps to optimize the function. If not provided, the mean value is computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetX  QUEUE,PRE()
X          REAL
          END

SumOFSquaresX   REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 SumOFSquaresX = SS(StatSetX)        !call SS
```

SSXY (sum of squares for x and y)

SSXY(*dataset* [,*meanX*] [,*meanY*])

SSXY Returns the sum of the products of the differences between each data point and its associated (either X or Y) mean value.

dataset The label of a QUEUE with its first two component fields defined as REAL. The first component contains the set of X values and the second component contains the set of Y values.

meanX A numeric constant or variable containing the average of the X values (optional).

meanY A numeric constant or variable containing the average of the Y values (optional).

SSXY computes the sum of the products of the differences between each data point and its associated (either X or Y) mean value. The function operates on the numeric sets defined by all the entries in the first two components of the designated QUEUE (*dataset*). The computation is a significant factor in linear regression formulas.

The optional parameters help to optimize the function. If not provided, the value(s) will be computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY QUEUE,PRE()
X          REAL
Y          REAL
          END
SumOfSquares   REAL

CODE
 STADAT:Id = STA:Id                 !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                               !load the QUEUE
   NEXT(StatisticsData)             !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetXY:X = STADAT:X         !load the QUEUE buffer
     StatSetXY:Y = STADAT:Y         !load the QUEUE buffer
     ADD(StatSetXY)                 !add to the QUEUE
   END
 END
 SumOfSquares = SSXY(StatSetXY)     !call SSXY
```

ST1 (student's t for a single mean)

ST1(*dataset*,*hypothesizedmean*)

ST1	Returns the Student's t value for a given data set and hypothesized mean value.
Dataset	The label of a QUEUE with its first component field defined as a REAL.
Hypothesizedmean	A numeric constant or variable representing a hypothesized mean value associated with the statistical test.

ST1 computes the Student's t value for a given data set and hypothesized mean value. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*). The returned t value is used in a statistical test of an hypothesis about a normally distributed population based on a small sample.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY  QUEUE,PRE()
X           REAL
           END
TValue     REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 TValue = ST1(StatSetX,HypothesizedMean) !call ST1
```

SDEVIATIONP (standard deviation of a population)

SDEVIATIONP(*dataset* [,*meanofset*])

SDEVIATIONP	Returns the standard deviation of the entire population of a numeric set.
dataset	The label of a QUEUE with its first component field defined as a REAL.
meanofset	A numeric constant or variable representing the average of the data set's values (optional).

SDEVIATIONP computes the standard deviation of a given population data set. SDEVIATIONS computes the standard deviation of a given sample data set. The 'Population' function call should be used only if the data set contains all of a population's values. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

The optional parameter helps to optimize the function. If not provided, the value is computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY   QUEUE,PRE()
X            REAL
            END
StandardDeviation    REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 StandardDeviation = SDEVIATIONP(StatSetX)     !call SDEVIATIONP
```

SDEVIATIONS (standard deviation of a sample)

SDEVIATIONS(*dataset* [,*meanofset*])

SDEVIATIONS	Returns the standard deviation of a sample of a numeric set.
dataset	The label of a QUEUE with its first component field defined as a REAL.
meanofset	A numeric constant or variable representing the average of the data set's values.

SDEVIATIONP computes the standard deviation of a given population data set. SDEVIATIONS computes the standard deviation of a given sample data set. The 'Sample' function call should be used only if the data set contains less than all of a population's values. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

The optional parameter helps to optimize the function. If not provided, the value is computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY  QUEUE,PRE()
X            REAL
           END
StandardDeviation    REAL


 CODE                                  !free the QUEUE
  CLEAR(STADAT:RECORD)                !clear the record buffer
  STADAT:Id = STA:Id                  !prime the record buffer
  STADAT:ItemNumber = 1
  SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
  LOOP                                !load the QUEUE
    NEXT(StatisticsData)              !read next record
    IF ERRORCODE() OR STADAT:Id NOT = STA:Id
      BREAK
    ELSE
      StatSetX:X = STADAT:X           !load the QUEUE buffer
      ADD(StatSetX)                   !add to the QUEUE
    END
  END
  StandardDeviation = SDEVIATIONS(StatSetX)     !call SDEVIATIONS
```

SUMM (summation of a set)

SUMM(*dataset*)

Returns the summation of all of a data set's values.

SUMM

dataset The label of a QUEUE with its first component field defined as a REAL.

SUMM computes the summation of all of a data set's values. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*). The computation is a significant factor in several statistical formulas.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY       QUEUE,PRE()
X                REAL
                 END
SummationOfSet      REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 SummationOfSet = SUMM(StatSetX)     !call SUMM
```

UPPERQUARTILE (upper quartile value of a set)

UPPERQUARTILE(*dataset*)

Returns the median of the upper half of an ordered numeric data set.

UPPERQUARTILE

Dataset The label of a QUEUE with its first component field defined as a REAL.

UPPERQUARTILE computes a value such that at most 75% of the set's values are less than the amount, and at most 25% of the set's values are greater than the amount. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

For example, given the data set: [1,2,3,4,5,6,7,8] the upper quartile value is 6.5.

In general, the UPPERQUARTILE function is only meaningful when the number of elements in the data set is large (ie. greater than 20).

See also:

LOWERQUARTILE, PERCENTILE.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY  QUEUE,PRE()
X           REAL
           END
UpperQuartileOfSet    REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X          !load the QUEUE buffer
     ADD(StatSetX)                  !add to the QUEUE
   END
 END
 UpperQuartileOfSet = UPPERQUARTILE(StatSetX)   !call UPPERQUARTILE
```

VARIANCEP (variance of a population)

VARIANCEP(*dataset* [,*meanofset*])

Returns the variance of the entire population of a numeric set.

VARIANCEP

dataset	The label of a QUEUE with its first component field defined as a REAL.
meanofset	A numeric constant or variable representing the average of the data set's values (optional).

VARIANCEP computes the variance of a given population data set. VARIANCES computes the variance of a given sample data set. The 'Population' function call should be used only if the data set contains all of a population's values. The function operates on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

The optional parameter helps to optimize the function. If not provided, the value is computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY       QUEUE,PRE()
X                REAL
                END
VarianceOfSet   REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 VarianceOfSet = VARIANCEP(StatSetX) !call VARIANCEP
```

VARIANCES (variance of a sample)

VARIANCES(*dataset* [,*meanofset*])

VARIANCES Returns the variance of a sample of a numeric set.

dataset The label of a QUEUE with its first component field defined as a REAL.

meanofset A numeric constant or variable representing the average of the data set's values (optional).

VARIANCES computes the variance of a given sample data set. VARIANCEP computes the variance of a given population data set. The 'Sample' function call should be used only if the data set contains less than all of a population's values. Both functions operate on on the numeric set defined by all the entries in the first component of the designated QUEUE (*dataset*).

The optional parameter helps to optimize the function. If not provided, the value is computed internally by the function.

Note:

The passed data set does not have to be sorted. The function copies the passed set. The passed data set is unchanged.

Return DataType: REAL

Example:

```
StatSetXY  QUEUE,PRE()
X           REAL
           END
VarianceOfSet      REAL

CODE
 FREE(StatSetX)                      !free the QUEUE
 CLEAR(STADAT:RECORD)                !clear the record buffer
 STADAT:Id = STA:Id                  !prime the record buffer
 STADAT:ItemNumber = 1
 SET(STADAT:KeyIdItemNumber,STADAT:KeyIdItemNumber !position file pointer
 LOOP                                !load the QUEUE
   NEXT(StatisticsData)              !read next record
   IF ERRORCODE() OR STADAT:Id NOT = STA:Id
     BREAK
   ELSE
     StatSetX:X = STADAT:X           !load the QUEUE buffer
     ADD(StatSetX)                   !add to the QUEUE
   END
 END
 VarianceOfSet = VARIANCES(StatSetX) !call VARIANCES
```

Index

_Print print a Crystal Report....244
Aborted Add/Change....171
Add ADO File Extension....7
Additional Libraries and Templates....223, 224
Additional Sort Fields Dialog....5
ADO Browse Box Select Button....84
ADO browse control....76
ADO Browse Procedure....176
ADO Browse Process Button....84
ADO Browse Refresh Code Template....58
ADO Browse Update Button control template....85
ADO Errors Object List Template....86
ADO Form....176
ADO Login Controls Control Template....89
ADO Process Extension....13
ADO Process Pause Button....86
ADO Process Procedure....176
ADO Report Procedure....176
ADO Save Button Control Template....87
ADO Support Template - Global Extension....28
ADO Support Templates....76, 84, 85, 86, 87, 89
AlignShortcutsToLeft position menu shortcuts to left....285
AlinkLookUp....278
AMORTIZE....248
application template....3
Application Wizard utility template....178
APR....250
AreShortcutsLeftAligned detect left aligned menu shortcuts....285
ASCII Print Button control template....89
ASCII Search Button control template....89
ASCII View control template....90
ASCII View in List box....91
auto-size....92
BLOBInControl Extension Template....20
Browse Box....92
Browse Customization....196
Browse Procedure Wizard....182
Browse QBE List Control Template....109
Browse Select button control template....112
Browse Update buttons control template....113
Browse Wizard....182
BrowseFuzzyMatching control template....103
BrowseGrid....105
BrowseNoRecords Button control template....108
BrowsePrintButton....106
BrowsePublishButton....107
BrowseQueryButton....157
BrowseToHTML....107
BrowseToolbarControl....112
BrowseToolboxButton....112
BrowseViewButton....116
Business Math Library248, 250, 251, 252, 253, 254, 255, 257, 259, 261, 263, 264, 265, 267, 269, 270, 271, 273, 307, 308, 309, 310, 311, 312, 313, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325
Button Customization....198
Calendar Button - Classes Tab....118
Calendar Button - General Tab....117
Calendar Button Control Template....116
Call a Procedure Extended code template....59
Call Procedure As Lookup code template....60
CallABCMethod....58
Cancel Button control template....120
ChangeRecord....1
CleanCloseDown Global Extension....21
Close Button control template....120
Close Current Window code template....60
CloseHelp....279
code templates....56, 60, 66, 67, 69
COMPINT....251
CONTINT....252
Contract all....163
Control Customization....199

Control Model
Application Wizard 181
Control Model 181
control template 152, 160, 163
control templates 73, 89, 90, 112, 113, 120, 121, 123, 124, 171
Control Value Validation code template 60
Cooperative Threading 28
Crystal Reports 226
Crystal Reports Overview 224
Crystal Runtime Distribution 224
Crystal8 Class Properties 234
Crystal8 Methods 235
CwHHGlobal 276
cwHHProc 277
Data Dictionary
Printing 195
Data Dictionary 195
Date Stamp on a report 163
Date Time Display extension template 22
DAYS360 253
DbAuditing 23
DeleteRecord 1
Deleting records 171
Design Theme 218
Dictionary Print Wizard 195
Dictionary Print Wizard utility template 195
DisplayPopupMenu 61
DOS File Lookup control template 121
DynamicImage 123
Edit in Place Support 113
EIP Template Support 113
Embedded Source 56
Expand all 163
Export to XML code template 62
ExtendProgressWindow 24
extension template 4, 22, 32
FACTORIAL 306
Field Lookup Button control template 123
fields
printing 195
fields 195
File Drop Combo control template 129
File Drop control template 124
files
printing 195
files 195
FileUserTags 25
Form Customization 200
Form Wizard 184
FormVcrControls extension template 25
FormVCRForm Control 134
Frame Browse Control 137
Frame Customization 202
FrameExtension Template 26
FREQUENCY 307
FromXML code template 63
FV 254
GetCrystalFormulaPreview 227
GetCrystalQueryPreview 227
GetEndColor get ending gradient color 286
GetFlatMode get menu flat mode 287
GetFont get menu text font 287
GetFullDragSetting code template 64
GetHelpFileName 279
GetImage get image name assigned to menu part 288
GetIsVerticalGradient does menu part use a vertical gradient 289
GetOSVersion code template 64
GetStartColor get starting gradient color 290
GetTextColor get menu text color 291
GetTopicName 279
Global Cooperative Threading Extension 28
Global Rules 47
Global Rules - Controls 49
GlobalRequest 1

GlobalResponse....1
Greenbar Effect - See Styles Tab....92
HALT - better alternative....21
HasCancelButton....237
HasCloseButton....237
HasExportButton....238
HasLaunchButton....238
HasNavigationControls....239
HasPrintButton....239
HasPrintSetupButton....240
HasProgressControls....240
HasRefreshButton....241
HasSearchButton....241
HasZoomControl....242
Help Button control template....139
Help ID Utility....194
HLP IDs - List....194
Hot Fields....92, 124, 129, 134
HTML Help - Overview....274
HTML Help Templates....275
Import From XML code template....65
Init initialize Crystal8 object....242
Init initialize the MenuStyleManager object....292
Initiate Thread code template....66
InitMenuInterface initialize menu properties....293
InsertRecord....1
Inserts....171
IRR....255
Keys
printing....195
Keys....195
KeyWordLookUp....280
Kill shut down Crystal8 object....242
Label Wizard utility template....187
Local Business Rules....52
Local Business Rules - Hot Fields....52
Local Business Rules - Locally Defined....53
Local Business Rules - Special Controls....54
Local Business Rules - Used Controls....54
Local Business Rules - Used Global Rules....55
LocalRequest....1
LocalResponse....1
Locators....92
lookup
field....123
lookup....123
Lookup Non-Related Record code template....67
LOWERQUARTILE....308
Mask Description....121
MDI Synchronization
Global Extension....29
Procedure Extension....30
MDI Synchronization....29
MDI Synchronization....30
MEAN....309
MEDIAN....310
MenuFEQ Menu Field Equate....283
MenuHasGradient does menu part have gradient applied....294
MenuInterface IMenuInterface reference....295
MenuStyle IMenuStyle reference....296
MIDRANGE....311
MixColors....297
MODE....312
module template....3
MultiChildRelationTree Button control template....148
MultiChildRelationTree control template....140
MultiChildRelationTree Expand/Contract Buttons control template....149
MultiChildRelationTree Select Button control template....149
MultiChildRelationTree Update Buttons control template....150
NPV....257
OCX control....152
OLE control template....152
OriginalRequest....1

page number on a report....163
Pause Button Control template....153
Percentile....313
PERS....259
PMT....261
Preemptive Procedure Extension....30
PREFV....264
PREPERS....265
PREPMT....263
PREPV....269
PRERATE....267
Preview preview a Crystal Report....243
PreviewCrystalReport....228
PrintCrystalReport....231
Printing
Data Dictionary....195
Printing....195
procedure template....175
ProcedureCallTreeReport Utility Template....189
Process Control Customization....206
Process Customization....205
Process Transaction Frame Checkpoint code template....67
Process Transaction Frame extension template....31
Process Wizard utility template....190
ProcessRecord....1
ProcessReportQBEButton....154
program template....4
PV....270
QBE List....109
Query by Example....109
Query retrieve or set the SQL data query....244
RANGEOFSET....315
RATE....271
Rebase....35
Record Validation extension template....32
Refresh Window routine....2
Relation Tree Expand Contract buttons....163
Relation Tree Update buttons....163
relationships
printing....195
relationships....195
RelationTree control template....160
repetitive insert/add
field history key
ditto key
repeating entries into fields....171
repetitive insert/add....171
Report Additions Customization....215
Report Customization....207
Report Label Customization....212
Report Wizard....191
ReportChildFiles....33
ReportDateStamp control template....163
ReportPageNumber control template....163
ReportTimeStamp control template....164
Request and Response....1
resize strategy
for a single control....39
resize strategy....39
ResizeSetStrategy....68
Rich Text Templates....305
RTF Popup Menu translation....165
RTFStatusBar Control Template....164
RTFTextControl Control Template....165
RTFToolBar Control Template....169
RunCommandLineProc....34
RVALUE....316
Save As Theme....216
Save Button control template....171
Save Button Transaction Frame extension template....34
Save Theme....216
SaveRecord....1
SDEVIATIONP....320
SDEVIATIONS....321
Select Embed Type....56

Select Utility Template 177
SelectionFormula retireve or set the Crystal formula ..245
SelectRecord 1
SelectToolbarTarget 69
SetABCProperty 69
SetColor specify color for menu part 298
SetControlStrategy 39
SetCrystalFormulaPreview 232
SetCrystalQueryPreview 233
SetFlatMode set menu flat mode 299
SetFont set menu text font 300
SetFullDragSetting code template 70
SetHelpFileName 280
SetImage apply image to menu part 301
SetProperty code template 70
SetTextColor set menu text color 302
SetThemeColors 303
SetTopicName 280
SetVerticalGradient set vertical color gradient 304
ShowDocumentTips show tips on docuement in the preview window 245
ShowIndex 281
ShowReportControls show print controls 246
ShowSearch 281
ShowTOC 281
ShowToolbarTips show tips on preview window toolbar 246
ShowTopic code template 281
SIMPINT 273
Sort Columns - Extended Options 92
Sort Header - Extended Options 92
Sort Order Button control template 173
Sort Order Customization 217
Sort Order Drop List control template 174
SQL File Systems
Special Application Features for 178
SQL File Systems 178
SS 317
SSXY 318
ST1 319
Styles - Browse Box 92
SUMM 322
Template Utility 195
Templates 22, 32, 60, 66, 67, 89, 90, 112, 113, 120, 121, 123, 124, 171
Theme Maintenance Wizard 220
time stamp on a report 164
toolbar control 137
Toolbar Model
Button Model 181
Toolbar Model 181
ToXML code template 71
TransactionManager template support 31, 67
Tree control 160
UPPERQUARTILE 323
Utility Template 195
Utility Templates 177
VARIANCEP 324
VARIANCES 325
VCR Form Buttons 134
Version Resource extension template 36
View Only Button control template 116
ViewRecord 1
ViewXML code template 72
Window Customization 221
Window Wizard utility template 193
WindowResize extension template 39
Wise-Generate Wise Installation Script 44
Wizard - Browse 182
Wizard-Control model 181
Wizards 177, 178, 184, 191, 195
Write Help IDs Utility Template 194
XML 62, 63, 65, 71, 72