

New Features

The following pages are intended as a supplement to the C55 documentation set. This document gathers all of the new features included in the C55.01 update.

The redirection file used for C55.01 has been renamed from Clarion5.RED to C55EE.RED or C55PE.RED, depending on your product version. Any changes made to the Clarion5.RED should also be made in C55EE.RED or C55PE.RED.

Language Reference

FONTDIALOG and FONTDIALOGA have been updated to allow the last parameter be set to four (4), add true type only fonts, and eight (8), add fixed pitch only fonts to the FontDialog font selection. The full text for these two functions appear in this document.

The LIST control's FROM attribute syntax has been updated to distinguish the strings displayed in a list and the values that are assigned to the control's USE variable upon selecting another entry in the LIST. See the FROM attribute documentation in this document.

A new function has been added to the language. STRPOS is similar to the existing MATCH function but returns the position in the searched substring as compared to MATCH which returns a TRUE or FALSE denoting whether the search string was found. The full text for this new function appears in this document.

ABC Libraries

The Crystal8 class documentation has been added to the ABC Library Reference, Volume 1. The complete class documentation can also be found in this document.

A new method has been added to the cwRTF class. It is called SetTip and is used to rename the tooltips for toolbars and popup menu entries.

The cwRTF.SelectText method has been updated to allow parameters of 0, -1, to allow selection of all text in the rich text control.

Programmer's Guide

A new DDE Service has been added. This service allows a call to Clarion to Unregister a template chain. See the *Unregister a Template Chain* documentation in this document.

User's Guide

A new tab has been added to the Editor Option dialog. This tab has prompts to set the editor font to any fixed pitch font. Choose from Default, System, or Custom to select a customized font setting.

LANGUAGE REFERENCE MANUAL

The following Language Reference Manual section contains the updated documentation for FONTDIALOG, FONTDIALOGA, FROM, and the new STRPOS funtion.

FONTDIALOG (return chosen font)

FONTDIALOG([*title*] ,*typeface* [*size*] [*color*] [*style*] [,*added*])

FONTDIALOG	Displays the standard Windows font choice dialog box to allow the user to choose a font.
<i>title</i>	A string constant or variable containing the title to place on the font choice dialog. If omitted, a default <i>title</i> is supplied by Windows.
<i>typeface</i>	A string variable to receive the name of the chosen font.
<i>size</i>	An integer variable to receive the size (in points) of the chosen font.
<i>color</i>	A LONG integer variable to receive the red, green, and blue values for the color of the chosen font in the low-order three bytes.
<i>style</i>	An integer variable to receive the strike weight and style of the chosen font.
<i>added</i>	An integer constant or variable that specifies adding screen or printer fonts, or both, to the list of available fonts. Zero (0) adds screen fonts, one (1) adds printer fonts, two (2) adds both, four (4) adds true type only fonts, and eight (8) adds fixed pitch only fonts. If omitted, only Windows registered fonts are listed.

The **FONTDIALOG** procedure displays the Windows standard font choice dialog box to allow the user to choose a font. When called, any values in the parameters set the default font values presented to the user in the font choice dialog. They also receive the user's choice when the user presses the Ok button on the dialog. FONTDIALOG returns zero (0) if the user pressed the Cancel button, or one (1) if the user pressed the Ok button.

Return Data Type: **SHORT**

Example:

```

MDIChild1  WINDOW('View Text File'),AT(0,0,320,200),MDI,SYSTEM,HVSCROLL
           !window controls
           END
Typeface   STRING(31)
FontSize   LONG
FontColor  LONG
FontStyle  LONG
            CODE
OPEN(MDIChild1)                      !open the window
IF FONTDIALOG('Choose Display Font',Typeface,FontSize,FontColor,FontStyle,0)
  SETFONT(0,Typeface,FontSize,FontColor,FontStyle) !Set window font
ELSE
  SETFONT(0,'Arial',12)                 !Set default font
END
ACCEPT
!Window handling code
END

```

FONTDIALOGA (return chosen font and character set)

FONTDIALOGA([*title*] ,*typeface* [*size*] [*color*] [*style*] [*charset*] [*added*])

FONTDIALOGA Displays the standard Windows font choice dialog box to allow the user to choose a font and a character set.

<i>title</i>	A string constant or variable containing the title to place on the font choice dialog. If omitted, a default <i>title</i> is supplied by Windows.
<i>typeface</i>	A string variable to receive the name of the chosen font.
<i>size</i>	An integer variable to receive the size (in points) of the chosen font.
<i>color</i>	A LONG integer variable to receive the red, green, and blue values for the color of the chosen font in the low-order three bytes.
<i>style</i>	An integer variable to receive the strike weight and style of the chosen font.
<i>charset</i>	A LONG integer variable to receive the character set value.
<i>added</i>	An integer constant or variable that specifies adding screen or printer fonts, or both, to the list of available fonts. Zero (0) adds screen fonts, one (1) adds printer fonts, two (2) adds both, four (4) adds true type only fonts, and eight (8) adds fixed pitch only fonts. If omitted, only Windows registered fonts are listed.

The **FONTDIALOGA** procedure displays the Windows standard font choice dialog box to allow the user to choose a font and character set. When called, any values in the parameters set the default font values presented to the user in the font choice dialog. They also receive the user's choice when the user presses the Ok button on the dialog. FONTDIALOGA returns zero (0) if the user pressed the Cancel button, or one (1) if the user pressed the Ok button.

Return Data Type: SHORT

Example:

```
Typeface      STRING(31)
FontSize     LONG
FontColor    LONG
FontStyle    LONG
CharSet      LONG
Added        SIGNED
CODE
OPEN(MDIChild1)           !open the window
IF FONTDIALOGA('Choose Display Font',Typeface,FontSize,FontColor,FontStyle,CharSet,0)
  SETFONT(0,Typeface,FontSize,FontColor,FontStyle,CharSet) !Set window font
ELSE
  SETFONT(0,'Arial',12)          !Set default font
END
ACCEPT
!Window handling code
END
```

or

```
Typeface      STRING(31)
FontSize     LONG
FontColor    LONG
FontStyle    LONG
CharSet      LONG
Added        SIGNED
CODE
OPEN(MDIChild1)           !open the window
IF FONTDIALOGA('Choose Display Font',Typeface,FontSize,FontColor,FontStyle,|
  CharSet,FONT:TrueTypeOnly)
  SETFONT(0,Typeface,FontSize,FontColor,FontStyle,CharSet) !Set window font
ELSE
  SETFONT(0,'Arial',12)          !Set default font
END
ACCEPT
!Window handling code
END
```

FROM (set listbox data source)

FROM(*source*)

FROM Specifies the source of the data displayed or printed in a LIST control.

source The label of a QUEUE or field within a QUEUE, or a string constant or variable (normally a GROUP) containing the data items to display or print in the LIST. If the QUEUE has been dynamically created with NEW, the corresponding DISPOSE must come after the window has been closed.

The **FROM** attribute (PROP:From, write-only) specifies the source of the data elements displayed in a LIST, COMBO, or SPIN control, or printed in a LIST control.

If a string constant is specified as the *source*, individual data elements must be delimited by a vertical bar (|) character. To include a vertical bar as part of one data element, place two adjacent vertical bars in the string (||), and only one will be displayed. To indicate that an element is empty, place at least one blank space between the two vertical bars delimiting the elements (| |).

A value can be assigned to the LIST control's USE variable when a new entry is selected by specifying the value as part of the *source* string. The value is separated from the display string by a vertical bar character followed by a # (#). If the value must contain a vertical bar, place two adjacent vertical bars (||). The # character unless preceeded by a single vertical bar has no special effect. The # character also has no special effect if it is preceeded by a vertical bar that separates two data elements.

Window Usage

For a SPIN control, the *source* would usually be a QUEUE field or string. If the *source* is a QUEUE with multiple fields, only the first field is displayed in the SPIN.

For LIST and COMBO controls, the data elements are formatted for display according to the information in the FORMAT attribute. If the label of a QUEUE is specified as the *source*, all fields in the QUEUE are displayed as defined by the FORMAT attribute. If the label of one field in a QUEUE is specified as the *source*, only that field is displayed.

Report Usage

If the label of a QUEUE is specified as the *source*, all fields in the QUEUE are printed. If the label of one field in a QUEUE is specified as the *source*, only that field is printed. Only the current QUEUE entry in the queue's data buffer is printed in the LIST. If a string constant or variable is specified as

the *source*, the entire string (all entries in the vertical bar delimited list of data elements) is printed in the LIST. The data elements are formatted for printing in the LIST according to the information in the FORMAT attribute.

Example:

```

TD      QUEUE,AUTO
FName   STRING(20)
LName   STRING(20)
Init    STRING(4)
Wage    REAL
END

CustRpt REPORT,AT(1000,1000,6500,9000),THOUS
CustDetail DETAIL,AT(0,0,6500,1000)
        LIST,AT(0,34,366,146),FORMAT('80L80L16L60L'),FROM(TD),USE(?Show1)
        LIST,AT(0,200,100,146),FORMAT('80L'),FROM(Fname),USE(?Show2)
END

Que1    QUEUE,PRE(Q1)
F1      LONG
F2      STRING(8)
END

Win1   WINDOW,AT(0,0,160,400)
        LIST,AT(120,0,20,20),USE(?L1),FROM(Que1),FORMAT('5C~List~15L~Box~'),COLUMN
        COMBO(@S8),AT(120,120,20,20),USE(?C1),FROM(Q1:F2)
        SPIN(@N8.2),AT(280,0,20,20),USE(SpinVar1),FROM(Q1:F1)
        SPIN(@S4),AT(280,0,20,20),USE(SpinVar2),FROM('Mr.|Mrs.|Ms.|Dr.')
END

```

Example:

```

PROGRAM

MAP
END

S      STRING(2)

W      WINDOW('Caption'),AT(,,104,123),FONT('MS Sans Serif',8,,FONT:regular), |
        SYSTEM,GRAY,AUTO
        LIST,AT(20,14,60,10),USE(S),DROP(7),FROM('Sunday|#Su|Monday|#Mo|' &|
        'Tuesday|#Tu|Wednesday|#We|Thursday|#Th|Friday|#Fr|Saturday|#Sa')
        BUTTON('OK'),AT(31,91,45,14),USE(?OK),STD(STD:Close)
END

CODE
S = 'Tu'
OPEN (W)

ACCEPT
END

CLOSE (W)
MESSAGE (S)

```

STRPOS (return matching value position)

STRPOS(*first*, *second* [, *mode*])

STRPOS	Returns the starting position of a substring based on a comparison of the first two parameters passed.
<i>first</i>	A string containing data to compare against the <i>second</i> parameter.
<i>second</i>	A string containing data to compare against the <i>first</i> parameter.
<i>mode</i>	An integer constant or equate which specifies the method of comparison. If omitted, a wild card comparison is the default.

The **STRPOS** procedure returns the starting position where the *first* and *second* parameters match according to the comparison *mode* specified. The following *mode* value EQUATES are listed in EQUATES.CLW:

Match:Simple

A straightforward equivalence comparison (*first* = *second*), which is most useful when combined with Match:NoCase.

Match:Wild (default)

A wild card match with the *second* parameter containing the pattern that can contain 'asterisk (*) to match 0 or more of any character, and question mark (?) to match any single character.

Match:Regular

A regular expression match where the *second* parameter contains the regular expression. Repeated usage with the same regular expression value is optimized (to avoid re-compiling the expression).

Match:Soundex

A standard soundex comparison of the two strings, returning true if they have the same soundex value.

Match:NoCase

Add to the *mode* for a case insensitive match (except Soundex).

Regular Expression Operators

Regular expressions are used to describe patterns in text. The following characters are regular expression operators (or metacharacters) used to increase the power and versatility of regular expressions.

- ^ Caret matches the beginning of the string. For example:

^@chapter

matches the “@chapter” at the beginning of a string.

- \$ Dollar sign is similar to the caret, but it matches only at the end of a string. For example:

p\$

matches a record that ends with a p.

- . Period matches any single character except a newline. For example:

.P

matches any single character followed by a P in a string. Using concatenation we can make regular expressions like `U.A', which matches any three-character sequence that begins with `U' and ends with `A'.

- [...] This is called a character set. It matches any one of the characters that are enclosed in the square brackets. For example:

[MVX]

matches any one of the characters M, V, or X in a string. Ranges of characters are indicated by using a hyphen between the beginning and ending characters, and enclosing the whole thing in brackets. For example:

[0-9]

matches any digit. To match `-', write it as `---', which is a range containing only `+'. You may also give `-' as the first or last character in the set. To match `^', put it anywhere except as the first character of a set. To match a `]', make it the first character in the set. For example:

[]d^]

matches either `]', `d' or `^'.

[^ ...]

This is a complemented character set. The first character after the [must be a ^. It matches any characters except those in the square brackets (or new line). For example:

[^0-9]

matches any character that is not a digit.

- | Vertical bar is the alternation operator and it is used to specify alternatives. For example:

^P|[0-9]

matches any string that matches either ^P or [0-9]. This means it matches any string that contains a digit or starts with P. The alternation applies to the largest possible regexps on either side.

{...}

Parentheses are used for grouping in regular expressions as in arithmetic. They can be used to concatenate regular expressions containing the alternation operator, |.

- * Asterisk means that the preceding regular expression is to be repeated as many times as possible to find a match. For example:

`ph*`

applies the * symbol to the preceding h and looks for matches to one p followed by any number of h's. This will also match just p if no h's are present. The * repeats the smallest possible preceding expression (use parentheses if you wish to repeat a larger expression). It finds as many repetitions as possible. For example:

`(c[ad][ad]*r x)`

matches a string of the form (car x), (cdr x), (cadr x), and so on.

- + Plus sign is similar to *, but the preceding expression must be matched at least once. This means that:

`wh+y`

would match “why” and “whhy” but not “wy,” whereas wh*y would match all three of these strings. This is a simpler way of writing the last * example:

`(c[ad]+r x)`

- ? Question mark is similar to *, but the preceding expression can be matched once or not at all. For example:

`fe?d`

will match fed and fd, but nothing else.

- \ Backslash is used to suppress the special meaning of a character when matching. For example:

`\$`

matches the character \$.

In regular expressions, the *, +, and ? operators have the highest precedence, followed by concatenation, and finally by |.

Return Data Type: LONG

Example:

```
A STRING('abecdef')
B STRING('ABCDEF')
C STRING('c.*')
```

```
! STRPOS(A,B,Match:Simple+Match:NoCase)
! STRPOS(A,B,Match:Soundex)
! STRPOS(A,C)
```

Returns 1 - case insensitive match
Returns 1 - same soundex values
Returns 3 - wildcard match

ABC LIBRARY REFERENCE

The following ABC Library Reference section contains the documentation for the Crystal8 class as well as updated documentation for cwRTF methods, SetTip and SelectText.

CRYSTAL8

Overview

Seagate Software's Crystal Reports is one of the leading report writers delivering Windows reports. For more information on this product see Seagate Software at www.seagatesoftware.com.

Clarion's Crystal Report interface is comprised of templates, libraries, and DLLs that communicate with Seagate's Crystal Reports, version 8. 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 Seagate Software's Crystal Reports report writer. The DLL is accessed by the Crystal 8 Class. 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.

c55cr8.INC	Crystal Class Definition
c55cr8L.INC	Crystal Class Definition Local Compile
c55cr8.dll	Crystal DLL
c55cr8.lib	Crystal LIB
c55cr8L.lib	Crystal Local LIB

Crystal8 Methods

The Crystal8 class contains the following methods.

AllowPrompt (prompt for runtime parameter data)

AllowPrompt(| allowpromptflag |)

AllowPrompt	Allow report runtime prompting for Crystal Parameter fields.
<i>allowpromptflag</i>	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 previewd or printed to prompt for runtime paramater fields. These parameter fields must be defined in the Crystal report. This method returns a BYTE representing the value of *allowpromptdialog*.

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.
<i>candrilldown</i>	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) allow 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 suport. For more information on Crystal's drill down feature refer to the Crystal Report's 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 reports 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 Seagate 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 reports 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 reports 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 naviagtion 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 reports 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 reports 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 the window caption.

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 sucessfully 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 during 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 no frame under Windows 3.1 or a single pixel frame under Windows 96/98. This parameter is optional.

icon

A string constant, variable, EQUATE, or expression containing an icon filename. By specifying 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 background 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
```

ShowToolbarTips (show tips on preview window toolbar)

ShowToolbarTips(/ *show tooltips* |)

ShowToolbarTips Display tooltips in the preview window's toolbar.

show tooltips 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 *show tooltips*.

Return Data Type: BYTE

Example:

```
oCrystal8.ShowToolBarTips(1)
```

ShowDocumentTips (show tips on document in the preview window)

ShowDocumentTips(/ *show document tips* |)

ShowDocumentTips Display tooltips in the document being pre-viewed.

show document tips 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 *show document tips*.

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)
```

The SetTip method is a new method of the cwRTF class. The SelectText method has been updated.

SetTip (change tooltips of control in RTF toolbars)

SetTip(*tipname*, *tiptext*)

SetTip	Changes the tooltip of a control in the RTF toolbars.
<i>tipname</i>	A string constant or variable containing the control name to replace the tooltip for.
<i>tiptext</i>	A string contrant or variable containing the text to use as the tooltip for the control.

The **SetTip** method is used to change the default tooltip text for controls in the toolbars. This method must be called after the RTF control is initialized.

Available controls in the toolbar are:

NEW	OPEN	SAVE
PRINT	FIND	CUT
COPY	PASTE	UNDO
REDO		

Available controls in the format bar are:

FONTS	SIZE	BOLD
ITALIC	COLOR	LEFT
RIGHT	CENTER	BULLETS
UNDERLINE		

Example:

```
oRTF_RTFTextBox.SetToolTip('NEW', 'NUEVO')
```

SelectText (select characters)

SelectText(*startpos*, *endpos*)

GetText	Select a specific set of characters by character position.
<i>startpos</i>	An integer constant or variable that specifies the starting character position to select text from.
<i>endpos</i>	An integer constant or variable that specifies the ending character position to select text from.

The **SelectText** method is used select a specific set of characters by character position. The text is highlighted when it is selected. To select all text, specify 0 as the *startpos* and -1 as the *endpos*.

Example:

```
!Select text from position 1 to 20
oRTF_RTFTextBox.SelectText(1,20 )
!Select text from variable positions
oRTF_RTFTextBox.SelectText(loc:startpos,loc:endpos )
!Select all text
oRTF_RTFTextBox.SelectText(0,-1 )
```

PROGRAMMER'S GUIDE

The following Programmer's Guide section contains new documentation for a new DDE service, UnregisterTemplateChain.

UnRegistering a Template Class

```
DDEEXECUTE(Server,['UnRegisterTemplateChain(TemplateName,ScreenOutput)'])
```

DDEEXECUTE Send a command string to a DDE server.

Server A LONG containing the DDE client channel number.

UnRegisterTemplateChain

The service of unregistering a specified template set.

TemplateName The name of the template set to unregister.

ScreenOutput If one (1) or TRUE, screen output is displayed, if zero (0) or FALSE, screen display is suppressed. If omitted, screen output is suppressed.

This feature is primarily intended for add-on products for Clarion. By allowing a template set to be unregistered by a client application, template developers can add this functionality to the installation procedure.

Example:

```
Server      LONG
DDEErrorMsg CSTRING(300)
DDEErrorNum USHORT

CODE

SYSTEM{PROP:DDETimeOut} = 12000      ! Time out after two minutes
DDEEXECUTE(Server,['UnRegisterTemplateChain(MYTPL.TPL,0)'])
DO CheckDDError

CheckDDError   ROUTINE

DDEErrorMsg = ''
err# = ERRORCODE()
! Check for DDE error
IF err# > 600      ! ERRORCODE is DDE related
  IF err# = 603      !! DDEExecute Failed
    DDEREAD(Server, DDE:manual, 'GetErrorNum', DDEErrorNum)
    DDEREAD(Server, DDE:manual, 'GetErrorMsg', DDEErrorMsg)
    MESSAGE('Error ' & DDEErrorNum & ' : ' & CLIP(DDEErrorMsg))
  ELSIF err# = 605! Timeout Error
    MESSAGE('DDE timeout')
  ELSE
    MESSAGE(err#)
  END
END
```

USER'S GUIDE

The following User's Guide section contains updated documentation for Editor Options dialog.

The FONT tab in the Editor Options dialog has been added.

Editor Options Dialog

To open the **Editor Options** dialog, choose **Setup > Editor Options**. Select the corresponding tab to set specific Text Editor options.

Font

This tab provides the following font options.

Text Font

Any fixed pitch font can be used as an editor font. Choose from Default, System, or a customized font setting.

Default

The Default font is the IBM font set. This is the C55IBM.FON file with a default height, weight, and OEM character set.

System

The System default fixed font with default height, weight, and character set.

Custom

Choose a font, font style, and font size using a font selection window. The selected elements of the font are described on the **Font** tab. The selected font will use the character set specified for CLASYSTEMCHARSET in the environment's .ENV file.